Proceedings of the 2023 International Conference on Machine Learning and Cybernetics, 9-11 July, 2023

SOLVING THE VRP USING TRANSFORMER-BASED DEEP REINFORCEMENT LEARNING

XUE-LIAN REN[1,2], AI-XIANG(ANDY) CHEN[1,2*]

[1]School of Statistics and Mathematics, Guangdong University of Finance and Economics, Guangzhou, China
[2]Institute of Artificial Intelligence and Deep Learning, Guangzhou, China
E-MAIL: renxljy@163.com, cax413@163.com

2023 International Conference on Machine Learning and Cybernetics (ICMLC) | 979-8-3503-0378-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICMLC58545.2023.10327956

Abstract:

The Vehicle Routing Problem (VRP) is a well-known NP-hard problem, and finding fast and efficient algorithms for VRP has been a major research focus in the academic community. In recent years, the advancements in deep reinforcement learning have provided new possibilities for solving VRP. This paper proposes a novel approach for solving VRP using a Transformer-based deep reinforcement learning framework with an encoder-decoder structure. The encoder utilizes a Transformer model to encode the VRP problem, while the decoder incorporates the positional information of the nodes that have already been visited as input and generates a sequence of nodes to be visited as the output solution. Finally, the entire model is trained using reinforcement learning. Experimental results demonstrate that that the GAP value of our model in CRP20 has decreased to 0.705%, which is more stable than other models and has a much faster solving speed than the heuristic model.

Keywords:

Transformer; Deep learning; Reinforcement learning; Vehicle optimization problem; Location encoding

1. Introduction

Combinatorial optimization problems refer to finding the optimal solution for a problem's objective function under given constraints by considering various combinations. The Vehicle Routing Problem (VRP) is an NP-hard problem that has been a hot topic in both academia and industry for decades. Its objective is to plan vehicle routes for delivering goods to a fixed number of customers with optimal efficiency. These problems are typically solved using either exact algorithms or approximation algorithms. Exact algorithms aim to find the exact solution within a finite time, but their time complexity increases as the problem size grows. On the other hand, approximation algorithms can find near-optimal solutions in a reasonable amount of time but do not guarantee the optimal solution.

Traditional algorithms play a crucial role in solving combinatorial optimization problems. However, with the advancement of deep learning [1], researchers have started exploring the combination of deep neural networks and reinforcement learning algorithms to solve these problems. In this approach, deep learning models, often based on encoder-decoder architectures, are used to represent the problem state and action-value functions, while reinforcement learning algorithms are employed to learn the optimal policy. The advantages of deep reinforcement learning methods lie in their ability to learn problem-solving strategies directly from data through end-to-end learning, without specific assumptions or manual feature engineering. They also outperform many heuristic algorithms in terms of solution speed. However, these methods require a large amount of training data and computational resources, and the training process is typically time-consuming. Moreover, due to the complexity of combinatorial optimization problems, there is still significant room for improvement in the performance of deep reinforcement learning methods. In this study, a new model algorithm based on the Transformer model [2] and reinforcement learning is proposed to solve the VRP. The proposed model shows improved convergence speed and solution accuracy.

2. Related work

In 1985, Hopfield et al. [3] pioneered the integration of neural networks with combinatorial optimization problems and successfully applied neural networks to solve small-scale traveling salesman problems. However, it was not until 2015 when Vinyals et al. [4] introduced the Pointer Network model, which opened up new possibilities for using neural networks to tackle combinatorial optimization problems. This model

979-8-3503-0378-0/23/$31.00 ©2023 IEEE

proposed a novel encoder-decoder structure based on sequence-to-sequence (seq2seq) models and employed gradient optimization methods for end-to-end training, resulting in efficient training and strong generalization capabilities. However, the supervised training of this model is sensitive to the quantity and quality of the training samples, making it time-consuming to solve large-scale combinatorial optimization problems.

Bello et al. [5] made improvements to the aforementioned approach by modeling the combinatorial optimization problem as a Markov process, using the objective function as the reward signal, and training it with the REINFORCE algorithm [2]. Inspired by the Transformer model, Deudon et al. [6] adopted a similar encoder structure and used embeddings of the three most recent selected nodes as query vectors in the decoder. They continued training the model using reinforcement learning algorithms and optimized the initial solution through local search. This method [6] not only improved the quality of the solutions but also reduced the complexity of the model. Building upon this, KOOL et al. [7] extended the model by considering the nodes selected in the first step and the two most recent steps to make decisions for the next step. They also designed a more efficient Rollout baseline method to train the model, enabling it to solve various combinatorial optimization problems. Additionally, Bresson et al. [8] introduced positional encoding in the decoder, incorporating information about the visited nodes as input for the next step. Their model achieved higher accuracy compared to many heuristic methods.

3. VRP Solution Method Based on Deep Reinforcement Learning

This paper is based on deep reinforcement learning and improves the Transformer model to solve the CVRP (Capacitated Vehicle Routing Problem). Given a set $X = \{x_i, i = 0,1,2,\dots n\}$,where $i = 0$ represents the warehouse coordinates and the rest are customer .

coordinates, the vehicle capacity is denoted as D, and the demand for each customer node is $0 < d_j \leq D$. For any route k, $\sum_{j \in r_k} d_j \leq D$. Without loss of generality, this paper normalizes the customer demands as $\hat{d}_j = \frac{d_j}{D}$ and normalizes D as $\widehat{D}$. The input to the model includes node coordinates and customer demands $s = \{x_i, d_j, i = 0,1,2,\dots n, j = 1,2,\dots,n\}$, and it generates the output order π for visiting customer nodes. The CVRP is modeled as a Markov decision process, where the already visited customer sequence and the remaining vehicle capacity serve as the current state s, and the next node to be chosen represents the action to be taken. The negative of the sum of path lengths to visit each node, according to the policy π, is used as the reward. Here, the symbol $\|\cdot\|_2$ represents the l_2 norm, and the objective is to maximize the reward function.

$$R = -L(\pi|s) = -\sum_{j=1}^{r_k}\sum_{i=1}^{m}\left\|x_{\pi_{(i)}} - x_{\pi_{(i+1)}}\right\|_2 \quad (1)$$

The model proposed in this paper is based on modifying the decoder of the Transformer model. The model structure is shown in Figure 1. The following section will provide a detailed introduction to the model presented in this paper.

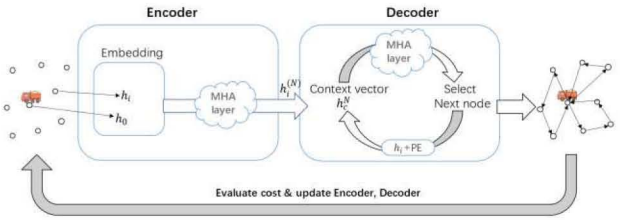

FIGURE 1. Network structure

3.1. Encoder

Based on the encoder design used in the Transformer architecture by Vaswani et al. [2], this paper adopts a similar encoder. To ensure that the generated node embeddings are not affected by the input order, positional encoding is no longer used in the encoder. The input is represented as $= \{x_i, d_j, i = 0,1,2,\dots n, j = 1,2,\dots,n\}$.

Given the input features, the Encoder performs a linear mapping using learned parameters W_x and b_x to obtain the initial node embeddings $H_i^{(0)}$. To differentiate between the warehouse node and the customer nodes, separate parameters W_{x_0} and b_{x_0} are used to calculate the initial node embedding vector $H_0^{(0)}$.

$$H_i^{(0)} = \begin{cases} W_{x_0}x_0 + b_{x_0} & i = 0 \\ W_x[x_i, \hat{d}_i] + b_x & i = 1,\dots,n \end{cases} \quad (2)$$

Then, N layers of the same attention module are used to update the node embedding vectors. Let $H_i^{(l)}$ represent the node embedding vector generated by layer $l \in \{1,\dots,N\}$, where $N = 3$ is the number of attention module layers, and the node embedding dimension is 128. Each attention module layer consists of two sub-layers: multi-head self-attention and a fully connected layer. Both sub-layers employ residual connections, which include a skip connection and batch normalization (BN). The implementation process is as follows:

$$\widehat{H}_i = BN^l\left(H_i^{(l-1)} + MHA_i^l\left(H_i^{(l-1)}\right)\right) \quad (3)$$

$$H_i^{(l)} = BN^l\left(\widehat{H}_i + FF^l\left(\widehat{H}_i\right)\right) \quad (4)$$

The term MHA represents Multi-Head Attention, which consists of 8 stacked self-attention mechanisms, i.e., $H = 8$, with a dimension of $\frac{d_h}{h}$=16. FF refers to a fully connected layer, composed of two fully connected neural network layers and one ReLU activation layer, with a dimension of 512.

Given the query vector $Q_i^{l,h} = W_Q^{l,h}H^{(l-1)}$, key vector $K_j^{l,h} = W_K^{l,h}H^{(l-1)}$, and value vector $V^{l,h} = W_V^{l,h}H^{(l-1)}$, The parameters $W_Q^{l,h}$, $W_K^{l,h}$, and $W_V^{l,h}$ are all trainable parameters. we can implement the multi-head attention mechanism as follows:

$$A^{l,h} = softmax\left(u_{ij}^{l,h}\right)V^{l,h} = softmax\left(\frac{Q_i^{l,h}K_j^{l,h^T}}{\sqrt{d_k}}\right)V^{l,h} \quad (5)$$

$$MHA(H^l) = \sum_{h=1}^{H=8} W_O^h A^{l,h} \quad (6)$$

Finally, by performing computations through the encoder, we obtain the final node embeddings $h_i^{(N)}$. The graph aggregation embedding $h_i^{(N)}$ is then obtained by taking the average of the final node embeddings, represented as $\bar{h}^{(N)} = \frac{1}{n}\sum_{i=1}^{n} h_i^{(N)}$.

3.2. Decoder

The decoding process is a sequential process where one customer is visited at a time, and the output is a sequence of nodes representing the order of visitation. The depot node can be visited multiple times. Assuming we have already visited t customers, we aim to predict the next node to be visited. The decoding process consists of the following four steps:

The decoding starts from the encoding of the previously visited city i_t.

$$H_t^{enc} = \left[h_{t_1}^{enc}, h_{t_2}^{enc}, \ldots, h_t^{enc}\right]$$

$$H_t = H_t^{enc} + PE_t$$

$$h_{t=0} = h_0$$

Here, PE_t represents the traditional positional encoding, which is used to sort the already visited customer nodes.

$$p = \begin{cases} \sin\left(\frac{i}{10000^{\frac{2j}{d}}}\right) & j \text{ is even} \\ \cos\left(\frac{i}{10000^{\frac{2j}{d}}}\right) & j \text{ is odd} \end{cases} \quad (7)$$

The self-attention mechanism is used to process the already visited nodes that have undergone positional encoding, resulting in the positional encoded node embeddings H_t^{pe}. Here, The parameters W_q, W_k and W_v are trainable parameters in this step.

$$H_t^{pe} = softmax\left(\frac{QK^{l^T}}{\sqrt{d}}\right)V^l \quad (8)$$

$$Q = H_t W_q \qquad K^l = H^{enc}W_k^{sh} \qquad V^l = H^{enc}W_v^l \quad (9)$$

Decoding - Step 3: In this stage, multi-head attention is used to obtain the final output H_t^{dec}. The context vectors used in this step are the positional encoded node embeddings h_t^{pe} of the previously visited customer nodes and the remaining vehicle capacity $\widehat{D}_t$. The context at $t = 1$ includes the encoded information of the depot (warehouse) and the maximum vehicle capacity.

$$h_t^{dec} = softmax\left(\frac{Q_{(C)}K^{h^T}}{\sqrt{d}}\right)V^h \quad (10)$$

$$Q_{(C)} = H_{(C)}W_q^h \qquad K^h = H^{enc}W_k^h \qquad V^h = H^{enc}W_v^h \quad (11)$$

$$H_{(C)} = \begin{cases} \left[\bar{h}^{enc}, h_t^{pe}, \widehat{D}_t\right] & t > 1 \\ \left[\bar{h}^{enc}, h_0^{enc}, \widehat{D}_t\right] & t = 1 \end{cases} \quad (12)$$

Finally, the self-attention mechanism is used to obtain the probability distribution p_t for the depot node and the unvisited customer nodes at the current time step. Based on this probability distribution, the next node to be visited is selected, and the demand of the visited customer node is set to zero. In this step, masking is applied as follows: masking the customer nodes with zero demand, masking all customer nodes if the remaining vehicle load is exactly 0, masking the customers whose demand exceeds the current vehicle load, and setting the logarithm of probability to negative infinity for infeasible solutions.

$$u_{(c)} = \begin{cases} C \cdot tanh\left(\frac{Q^o K^{o^T}}{\sqrt{d}}\right) & if\ j \notin \hat{\pi}_t \\ -\infty & otherwise \end{cases} \quad (13)$$

$$Q^o = H_t^{dec}W_q^o \quad K^o = H^{enc}W_k^o \quad (14)$$

$$p_t = softmax\left(u_{(c)}\right) \quad (15)$$

Here, W_q^o and W_k^o are trainable parameters, and C is a hyperparameter with a value of 10.

3.3. REINFORCE with greedy rollout baseline

Given an instance s, the encoder-decoder model generates a probability distribution, from which we can sample a solution π|s. To train the model, we define a loss function as follows:

Given an instance (current state) s, the encoder-decoder produces a probability distribution $p_\theta(\pi|s)$, from which we can sample a solution $\pi|s$ for state s. To train the model, a

loss function is defined:

$$\mathcal{L}(\theta|s) = \mathbb{E}_{p_\theta(\pi|s)} L(\pi) \quad (16)$$

The model in this paper utilizes the REINFORCE algorithm, which combines a Rollout baseline, to update the parameters through gradient descent.

$$\nabla L(\theta|s) = \mathbb{E}_{p_\theta(\pi|s)}[L(\pi) - b(s)) \nabla \log p_\theta(\pi|s)] \quad (17)$$

In this approach, the model $L(\pi)$ employs a random sampling strategy to select actions, while the baseline $b(s)$ uses a greedy algorithm to select actions. When the path length $L(\pi)$ of the policy π is lower than that of the baseline $b(s)$, the policy of $L(\pi)$ is updated to the new baseline policy.

4. Experiment

The implementation of the network model, algorithms, and training in this study was done using the PyTorch 1.1.7 deep learning framework in a Python 3.8 environment. The training process utilized a GPU, specifically the NVIDIA RTX 3060, to accelerate computations. The CRP20 and CRP50 problem instances were the focus of the training in this study.

4.1. Evaluation Metrics

The relative error between the optimal solution length $\hat{L}$ obtained using Gurobi and LKH3 and the approximate route length $L(\pi|s)$ generated by the model in this study was used as the criterion for assessment.

$$\text{GAP} = \frac{L(\pi|s) - \hat{L}}{\hat{L}} \quad (18)$$

4.2. Dataset and Settings

Considering the randomness in the distribution of customer nodes in real-life scenarios, the node coordinates in this paper's dataset are generated randomly following a uniform distribution. The number of customer nodes is set to 20 and 50, respectively. The customer demands are uniformly sampled from the discrete data {1, 2, 3, ..., 9}, where $D^{20} = 30$ and $D^{50} = 40$. The training is conducted for 100 epochs on the vehicle routing optimization problem with 20 and 50 customer nodes. For CRP20, each epoch consists of 2500 batches, with each batch containing 512 instances, resulting in a total of 1,280,000 problem instances. For VRP50, since the depot is visited multiple times, the solution length is greater than n, requiring more memory. Therefore, when $n = 50$, each batch uses 256 instance data. The Adam optimizer is utilized to update Equation (18) through gradient descent, with a learning rate set to 0.0001. The random number generator seed is set to seed=1234.

TABLE 1. My experimental results vs. baselines. The gap % is w.r.t. the best value across all methods.

method	CRP20			CRP50		
	Obj.	*GAP(%)*	*time*	*Obj.*	*GAP(%)*	*time*
Gurobi	6.10	0.00%	-	-	-	-
LKH3	6.14	0.58%	2h	10.38	0.00%	7h
RL	6.59	8.03%	-	11.39	9.78%	-
RL(beam 10)	6.40	4.92%	-	11.15	7.46%	-
Random CW	6.81	11.64%	-	12.25	18.07%	-
Random Sweep	7.08	16.07%	-	12.96	24.91%	-
AM	6.40	4.97%	1s	10.98	5.86%	3s
our	6.143	0.705%	2s	10.85	4.53%	8s

4.3. Experimental Results and Analysis

Using the path lengths obtained by the heuristic algorithm optimizers Gurobi and LKH3 as the benchmark, the optimal relative error GAP is calculated based on Equation (18). Comparing with other solving methods, the proposed model achieves a lower GAP. Table 1 presents the experimental results of the proposed model and other models on the same test sets of CRP20 and CRP50 problem instances. It can be observed that the GAP values are reduced to 0.705% for CRP20 and 4.53% for CRP50. Compared with heuristic

algorithms, the proposed model provides solutions that are very close to the heuristic solutions after training. Additionally, the proposed model is much faster than the heuristic models, requiring only 2 seconds to obtain solutions for the corresponding instances after training.

Figure 2 shows the optimization curves of Kool's model and the proposed model. The curve represents the minimum path lengths obtained by the models trained for each round on the same test set of 10,000 CRP20 instances. It can be seen that during the training process, Kool's model achieves a minimum path length of 6.328 on the test set, while the proposed model achieves a minimum path length of 6.143. Throughout the training process, the proposed model consistently outperforms Kool's model in terms of test results and demonstrates greater stability. Furthermore, by modifying the input data, the proposed model can also be applied to other combinatorial optimization problems such as TSP and OP.

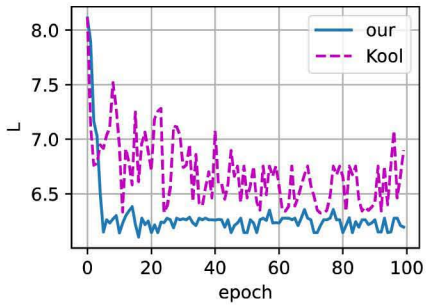

FIGURE 2. Held-out validation set optimality gap as a function of the number of periods for the attention model (AM) and my method with same baselines.

5. Conclusion

The proposed method in this paper for solving the vehicle routing optimization problem incorporates positional encoding, which takes into account the positional information of nodes in the sequence, effectively extracting the location information of visited nodes during the decoding process. Experimental results demonstrate that the model approaches the optimal solutions obtained by heuristic algorithms, accelerates the solving speed, and enhances the stability of the model. However, the model proposed in this paper focuses solely on traditional vehicle routing optimization problems. In reality, vehicle optimization problems often involve multiple objectives and constraints. Therefore, research on using deep reinforcement learning to solve multi-objective and multi-constraint vehicle optimization problems is an important direction to explore in the future.

Acknowledgements

This paper is partially supported by Graduate Course Construction Project "Modern Multivariate Statistical Analysis (English Course)" at Guangdong University of Finance and Economics, Chinese National Office for Philosophy and Social Sciences (19FTJB003), and National Bureau of Statistics of China (2016LY64).

References

[1] Aixiang chen. Deep learning. Tsinghua University press.2020

[2] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.

[3] John J Hopfield and David W Tank. "Neural" computation of decisions in optimization problems. Biological cybernetics, 52(3):141–152, 1985.

[4] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Advances in Neural Information Processing Systems, pp. 2692–2700, 2015.

[5] Bello I, Pham H, Le Q V, Norouzi M, Bengio S. Neural combinatorial optimization with reinforcement learning. In: 5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings, 2019.

[6] Deudon M, Cournut P, Lacoste A, Adulyasak Y, Rousseau LM. Learning heuristicsfor the tsp by policy gradient. In: International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research,Springer, 2018. 170-181.

[7] KOOL W, VAN HOOF H, WELLING M. Attention, learnto solve routing problems[J]. arXiv:1803.08475, 2018.

[8] Bresson X, Laurent T. The Transformer Network for the Traveling Salesman Problem[J]. arXiv:2103.03012, 2021.

[9] Dai H, Khalil EB, Zhang Y, Dilkina B, Song L. Learning combinatorial optimization algorithms over graphs. In: Advances in Neural Information Processing Systems, 2017. 6348-6358.

[10] Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2018. URL http://www. gurobi.com.

[11] Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems: Technical report. 2017.