Learning to Solve Vehicle Routing Problems with Soft Time Windows via Collaborative Transformer

Zengjian Yang[1], Junqing Li [2, 3,1, *], Xiaolong Chen[1]

[1] Shandong Normal University, Jinan, 250014, China
[2] Yunnan Normal University, Kunming, Yunnan, 650500, China
[3] HengXing University, Qingdao, 266199, China
lijunging@lcu-cs.com

Abstract. Over the past several years, there has been a rapid evolution in harnessing advanced deep reinforcement learning techniques to address challenges, including but not limited to the Traveling Salesperson Problems (TSPs) and the Vehicle Routing Problems (VRPs). However, the effectiveness of existing deep architectures for the Vehicle Routing Problem with Soft Time Windows (VRPSTW) is compromised by their integration of node and positional information into a single unified representation. In this article, we design a novel Collaborative Transformer framework based on deep reinforcement learning architecture to learn the node features(e.g., locations, time window) and positional features separately to avoid incompatible correlations, so as to improve the learning ability. During training, we leverage the Proximal Policy Optimization(PPO) algorithm to update the parameters of the model. This CT architecture serves as the policy network in the PPO framework. Tested on three datasets with customer points of 20, 50, and 100 respectively, experiments show that our method outperforms existing DRL architecture, showcasing its effectiveness in solving the given task.

Keywords: Vehicle routing problem with soft time windows, Transformer, Deep reinforcement learning.

1 Introduction

The foundation of intelligent transportation systems (ITS) relies on the aspiration to enhance transportation through technology, with a recent emphasis on Big Data algorithms. Continuous development of ITS and vehicle routing planning has become a key problem study in smart cities [1]. Overtime with the evolution of supply chain management approaches the delivery times that customers demand are more precise in last mile parcel delivery for smart cities [2]. To optimize the aforementioned requirements, we propose a method of collaborative attention mechanism to solve vehicle routing problem with soft time windows (VRPSTW).

The vehicle routing problem (VRP) is a typical NP-hard problem in combinatorial optimization and has been the subject of combinatorial optimization problems (COP) for several decades. Recently, there has been a growing effort to utilize DRL to address more complex versions of the VRP. An emerging trend involves integrating temporal constraints, such as time windows [3], [4], as customers typically prefer service within specific time frames, deviations from which may lead to dissatisfaction. Another notable trend involves extending from a single vehicle (or salesman) to multiple ones [5], reflecting the common practice of providing delivery services to customers in real-life situations. However, few studies have explored the application of DRL to simultaneously tackle both time windows and multiple vehicles.

The vehicle routing problem with soft time windows (VRPSTW) is a variant of the vehicle routing problem(VRP) where vehicles are tasked with delivering goods or providing services to a set of customers while respecting both capacity constraints and time window constraints [6]. In VRPSTW, the time windows are considered soft, meaning that there is flexibility in meeting the time constraints, but deviations from the specified time windows incur penalties. The objective of VRPSTW is to minimize the total cost, which typically includes transportation costs, penalty costs for violating time windows, and possibly other relevant costs.

Recent years have witnessed a surge of interest in applying deep reinforcement learning (DRL) techniques to tackle combinatorial optimization problems. Combinatorial optimization problems, such as the Traveling Salesman Problem (TSP), the Vehicle Routing Problem (VRP), and the Knapsack Problem, are prevalent across various domains, including logistics, operations research, and manufacturing. Traditionally, the mainstream methods for these problems can be classified into two categories: exact methods and heuristic methods [7]. While these methods can effectively solve VRP, they all have certain drawbacks, such as computational complexity and inability to efficiently handle large problem instances. However, DRL offers a novel paradigm that leverages the power of deep neural networks and reinforcement learning to discover effective solutions in a data-driven manner.

One seminal work in this field is "Neural Combinatorial Optimization with Reinforcement Learning" by Bello et al.[8], which proposed a framework for solving combinatorial optimization problems using DRL, and demonstrated the effectiveness of their approach on tasks such as the TSP and the VRP. Another significant contribution is the Attention Model (AM) by Kool et al. [9], which is regarded as the first successful VRP model based on Transformer and introduces the Pointer Network (PN). The AM achieves state-of-the-art performance on various combinatorial optimization benchmarks, including TSP and VRP instances. Furthermore, "Reinforcement Learning for Solving the Vehicle Routing Problem" by Nazari et al.[10] investigates the effectiveness of different DRL algorithms, including Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO)[16], for solving VRP instances. The study provides insights into the performance of various algorithms and their scalability to large-scale problem instances. In addition, Li et al.[11] proposed a novel encoder－refiner－decoder structure for enhancing the performance of neural construction methods in solving Vehicle Routing Problems.

In this paper, we propose a Collaborative Transformer (CT) architecture based on deep reinforcement learning to address the VRPSTW, a more challenging yet highly practical variant of the classic VRP, thereby offering a versatile and effective solution for optimizing vehicle routing in complex and dynamic logistical settings.

2 Problem Description and Formulation

This paper studies the vehicle routing problem with soft time windows (VRPSTW). This this section presents a gentle introduction to the VRP along with its mathematical formulation.

The objective in solving the vehicle routing problem with soft time windows (VRPTW) is to minimize the total cost associated with delivering goods or providing services to a set of customers, considering both transportation costs and penalties incurred for deviations from the specified time windows, while ensuring that all customer demands are met and vehicle capacity constraints are satisfied. The assumptions of VRPSTW are as follows:

(1)All customers must be visited exactly once.

(2)As a vehicle deployment center, the depot is the origin and destination of the vehicle.

(3)Set the vehicle speed to be a constant value.

(4)The satisfaction of customers with the service can be adversely affected when vehicles arrive outside the designated time windows as per the customers' requirements.

The VRPSTW is defined on a graph with a set of n nodes $\mathrm{C} = \{c_1, \ldots, c_n\}$, Each customer $\mathrm{c_i} \triangleq (x_i, y_i, d_i, e_i, l_i)$ has a 2D location (x_i, y_i), demand d_i and time window (e_i, l_i), where e_i and $l_i(e_i \leq l_i)$ denote the early and later time, respectively. The depot is represented as $N_0 \triangleq (x_d, y_d, 0, 0, \infty)$. Each vehicle departs from the depot, serves a subset of customers along a route, and finally returns to the depot. Arriving earlier than e_i or later than l_i of vehicles results in dissatisfaction.

The symbols used in the VRPSTW model are formulated as follows in **错误!未找到引用源。**:

Table 1. The symbols used in the VRPSTW model.

Symbol	Implication
N	Set of depot and customer.
N_0	The depot.
C	Set of customer nodes.
Q	Capacity of vehicles.
i	Indicator for node *i*.
j	Indicator for node *j*.
d_i	Set of delivery demands of customer *i*.
e_i	Earliest time window for customer *i*.
l_i	Latest time window for customer *i*.
$d_{i,j}$	Euclidean distance between nodes *i* and *j*.
t_i	Arrival time of vehicle reach node *i*.

Decision variables:

$$x_{i,j} = \begin{cases} 1, \text{vehicle starts from node } i \text{ to node } j \\ 0, \text{otherwise} \end{cases} \tag{1}$$

$$y_i = \begin{cases} 1, \text{vehicle serves customer } i \\ 0, \text{otherwise} \end{cases} \tag{2}$$

Objective function:

$$min(D(\delta)+U(\delta)) \tag{3}$$

$$D(\delta) = \sum_{i\in \mathrm{N}} \sum_{j\in \mathrm{N}} d_{i,j} x_{i,j} \tag{4}$$

$$U(\delta) = \begin{cases} t_i - e_i, & t_i < e_i \\ 0, & e_i <= a_i <= l_i \\ l_i - t_i, & t_i > l_i \end{cases} \tag{5}$$

Subjective to:

$$\sum_{j\in \mathrm{N}} x_{i,j} = \sum_{j\in \mathrm{N}} x_{j,i}, \forall i \in \mathrm{C} \tag{6}$$

$$\sum_{i\in \mathrm{N}} x_{i,j} = 1, \forall j \in \mathrm{C} \tag{7}$$

$$\sum_{i\in \mathrm{C}} x_{i,\mathrm{N}_0} = \sum_{j\in \mathrm{C}} x_{\mathrm{N}_0,j} \tag{8}$$

$$\sum_{i\in \mathrm{C}} d_i y_i \le \mathrm{Q} \tag{9}$$

$$x_{i,j} \in \{0,1\}, \forall i, j \in \mathrm{N} \tag{10}$$

$$y_i \in \{0,1\}, \forall i \in \mathrm{C} \tag{11}$$

Equation (3) expresses the objective of this paper, minimizing a hybrid cost of length and customer dissatisfaction. Equations (4) and (5) represent the distance traveled by all vehicles in the process of serving customers and penalties incurred for deviations from the specified time windows, respectively. Constraint (6) guarantees that the vehicle goes to the next destination after serving the customer. Constraint (7) ensures that a customer can only be accessed once. Constraint (8) can ensure that the depot is the origin and destination of the vehicles. Constraint (9) ensures that the sum of the demand in every sub-route does not exceed the vehicle capacity. Constraints (10) and (11) limit the scope of the decision variable.

3 Collaborative Transformer for VRPSTW

To efficiently address VRPSTW, we introduce a DRL framework aimed at learning improved solutions for routing problems. Our approach involves designing a transformer-based deep architecture called Collaborative Transformer (CT) as the policy network, which guides the selection of the next solution. The CT model capitalizes on the utilization of two distinct embeddings to encode a solution within the context of the VRPTW. This section presents the architecture of our collaborative transformer model. The framework of our collaborative Transformer is presented in Figure 1. The model's workflow proceeds as follows: In the encoder architecture described, each embedding undergoes individual computation of self-attention correlations. To enhance the effectiveness of attention correlations, a collaborative attention mechanism is introduced[12]. This mechanism facilitates the mutual exploitation of attention correlations between PEFs and NEFs, allowing each aspect to utilize the attention patterns of the other for more information. Following the encoding stage, the decoder synthesizes these encoded information into a final action distribution.

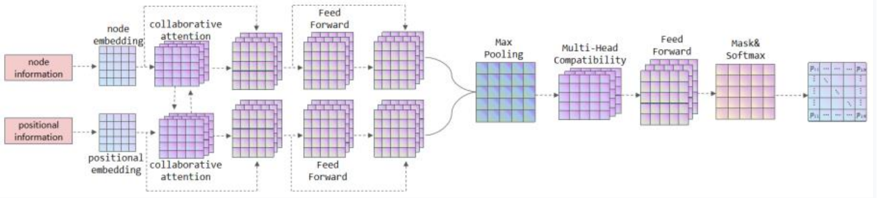


Fig.1. Architecture of our policy network

3.1 Embeddings

We divide the embeddings component of our model into two parts, namely the node embedding responsible and the positional embedding, where node embedding consists of a linear layer for extracting node information and outputs node feature embeddings(NFEs). Positional embedding consists of cyclic positional encoding (CPE) layer based on Ma et al.[12] that outputs positional feature embeddings(PFEs). Node information x_i represents coordinates, node demands, and time windows of node i, while positional information p_i denotes the visiting order of the customers within the sub-routes.

3.2 Encoder

The encoder in our model is composed of a stack of L = 3 identical layers, has its own parameters, and does the same computation. Each layer consists of four sub-layers, namely the collaborative-attention(CA) sub-layer, skip connection[13] and normalization[14] sub-layer, and feed-forward network(FFN) sub-layer. Different from the original Transformer[15], we use collaborative-attention to replace the multi-head attention in the original Transformer, the encoding process is shown in Eq.(12) and Eq.(13), where l represents the number of layers. The encoder receives both sets of NFEs and PFEs as input, facilitating collaboration and achieving information exchange between two embeddings.

$$\begin{aligned} NFE_i^{(l)} &= \mathrm{Norm}\left(NFE_i' + \mathrm{FFN}_h^{(l)}(NFE_i')\right), \\ NFE_i' &= \mathrm{Norm}\left(NFE_i^{(l-1)} + \mathrm{CA}_h^{(l)}\left(NFE_i^{(l-1)}\right)\right), \end{aligned} \tag{12}$$

$$\begin{aligned} PFE_i^{(l)} &= \mathrm{Norm}\left(PFE_i' + \mathrm{FFA}_g^{(l)}(PFE_i')\right), \\ PFE_i' &= \mathrm{Norm}\left(PFE_i^{(l-1)} + \mathrm{CA}_g^{(l)}\left(PFE_i^{(l-1)}\right)\right), \end{aligned} \tag{13}$$

The specific calculation process of the CA sub-layer is as follows:

1) First, taking both sets of embeddings NFEs and PFEs as input, calculating the self-attention correlation from both embeddings, as shown in Eq.(14), where W_h^Q, W_h^K, W_g^Q and $W_g^K \in R^{dim*d_k}$ are all trainable parameters used for computing queries and keys of NFES $\{NFE_i\}_{i=1}^N$ and PFES $\{PFE_i\}_{i=1}^N$, respectively. Then, the obtained $\alpha_{i,j}^h$ and $\alpha_{i,j}^g$ are further enhanced using the softmax function to get $\tilde{\alpha}_{i,j}^h$ and $\tilde{\alpha}_{i,j}^g$, respectively.

$$\alpha_{i,j}^h = \frac{1}{\sqrt{d_k}}\left(NFE_i W_h^Q\right)\left(NFE_j W_h^K\right)^T, \alpha_{i,j}^g = \frac{1}{\sqrt{d_k}}\left(PFE_i W_g^Q\right)\left(PFE_j W_g^K\right)^T, \tag{14}$$

2) Subsequently, similar to the model proposed by Ma et al.[12], the obtained correlations in the previous step are pairwise fused using a cross-aspect referential attention mechanism so that positional feature and node feature can be shared among each other. Finally, we concatenate the obtained values and referential values using the concat function, as shown in Eq.(15), where $W_h^v, W_g^v, W_h^{Vref}, W_g^{Vref}, W_h^o, W_g^o$ are also trainable parameters for computing values and referential values in each aspect and W_h^o, W_g^o are also trainable parameters.

$$
\begin{aligned}
out^{NFE_i} &= \mathrm{Concat}\left[\sum_{j=1}^{N}\tilde{\alpha}_{i,j}^{h}\left(NFE_j W_h^{v}\right), \sum_{j=1}^{N}\tilde{\alpha}_{i,j}^{g}\left(NFE_j W_h^{Vref}\right)\right] W_h^{o}, \\
out^{PFE_i} &= \mathrm{Concat}\left[\sum_{j=1}^{N}\tilde{\alpha}_{i,j}^{g}\left(PFE_j W_g^{v}\right), \sum_{j=1}^{N}\tilde{\alpha}_{i,j}^{h}\left(NFE_j W_g^{Vref}\right)\right] W_g^{o},
\end{aligned}
\tag{15}
$$

The AT layer is then followed by the FFN (Feed-Forward Network) layer of the encoder, it takes the obtained $out^{NFE_i^{(l)}}$ and $out^{PFE_i^{(l)}}$ as input, the parameters of $\mathrm{FFN_h}$ and $\mathrm{FFN_g}$ are different. This layer computers node-wise projections using a hidden sub-layer with dimension $d_{FF} = 64$ and the ReLU activation function. In essence, the FFN performs a linear transformation, a ReLU activation, and normalization, as shown in Eq.(16):

$$
\begin{aligned}
NFE_i^{(l)} &= \mathrm{Norm}[\mathrm{LN}(\mathrm{Relu}(\mathrm{LN}(out^{NFE_i^{(l)}}))) + out^{NFE_i^{(l)}}], \\
PFE_i^{(l)} &= \mathrm{Norm}[\mathrm{LN}(\mathrm{Relu}(\mathrm{LN}(out^{PFE_i^{(l)}}))) + out^{PFE_i^{(l)}}],
\end{aligned}
\tag{16}
$$

3.3 Decoder

For the decoder in our paper, we adopt the max pooling sub-layers, MHC sub-layer, and FFN sub-layer in Ma et al.[12]. Each of these sub-layer plays a crucial role in the decoding process. We first aggregate the outputs in the encoder into each respective representation by the max pooling sub-layer, as shown in Eq.(17), where $\mathrm{W_h^{Local}}, \mathrm{W_h^{Global}}, \mathrm{W_g^{Local}}, \mathrm{W_g^{Global}}$ are trainable parameters.

$$
\begin{aligned}
NFE_i &= NFE_i W_h^{Local} + \max\left[\{NFE_i\}_{i=1}^{N}\right] W_h^{Global}, \\
PFE_i &= PFE_i W_g^{Local} + \max\left[\{PFE_i\}_{i=1}^{N}\right] W_g^{Global},
\end{aligned}
\tag{17}
$$

Then, we compute the attention correlations for each Max-pooling layer result pair through the MHC sub-layer, which bears a significant resemblance to the attention mechanism in the encoder. Finally, they are further processed by the Feed-Forward Network(FFN) and Mask softmax layer, resulting in a probability distribution of size N × N. This distribution will serve as the action distribution for the ordering of nodes in the solution.

4 Reinforcement Learning Algorithm

In this paper, we used n-step Proximal Policy Optimization(PPO) algorithm[16] to train the network model, which is a variant of the actor-critic reinforcement learning algorithm. The Actor is responsible for generating actions, while the Critic estimates the value function, consistent with the fundamental Actor-Critic architecture. In particular, both the actor network and the policy network are represented by the same network denoted as π_θ. The critic network guides the updates of the actor network by evaluating the output of the actor denoted as v_ϕ. We train policy network π_θ and critic network v_ϕ using Adam optimizer. In particular, we use the advantage function to guide policy updates, PPO achieves effective and stable policy optimization within the Actor-Critic architecture. Our critic network v_ϕ is similar to that of actor in Ma et al.[12] as follows, consisting of multiple attention sub-layer, mean-pooling sub-layer, and feed-forward network used to assess the performance of the current policy.

Algorithm 1 n-Step PPO

```
Input: policy network $\pi_\theta$ with parameters θ; critic network $v_\phi$ with parameters ∅; E epochs , B batches; step limit T.
Output: θ and ∅ for the optimal actor network and critic network
for epoch=1 to E do
    Generate M problem instances randomly;
    For b=1 to B do
        Initialize random solutions $\{p_i\}$, set $s_0 = \{p_i\}$; $t \leftarrow 0$;
        while $t < T$ do
            $t_s = t$;
            while $t - t_s < n$ and not $(t == T)$ do
                Sample $a_t$ based on $\pi_\theta(a_t|s_t)$;
                receive reward $r_t$ and next state $s_{t+1}$;
                $t \leftarrow t + 1, \pi_{old} \leftarrow \pi_\theta, v_{old} \leftarrow v_\emptyset$
                for k=1 to K do
                    $R_{t+1} = v_\emptyset (s_{t+1})$;
                    for $i \in \{t, t-1, \ldots, t_s - 1\}$ do
                        $\mathbf{\hat{R}_i \leftarrow r_i + \gamma \hat{R}_{i+1}}$
                        $\hat{A}_i \leftarrow \hat{R}_i - v_\emptyset (s_i)$;
                    end for
                    $\theta \leftarrow \theta + \partial_\theta \nabla J_{PPO}(\theta)$;
                    $\emptyset \leftarrow \emptyset + \partial_\emptyset \nabla L_{BL}(\emptyset)$;
                end for
            end while
        end while
    end for
    $\partial_\theta \leftarrow \beta\, \partial_\theta, \partial_\emptyset \leftarrow \beta\, \partial_\emptyset$;
end for
```

As shown in Algorithm 1, we train π_θ and v_ϕ for multiple E epochs until predefined number of training steps is reached. For each batch, firstly, we randomly generate initial solutions for the instances and collect trajectories by interacting with the environment (line 7-10). Use the n-step returns to estimate advantages for each state-action pair in the batch. Afterwards, PPO performs K epochs of updates for network π_θ and v_ϕ by its objective, the PPO objective function is defined as in Equation (19), where the clip function is used to limit the range of policy variations to be within $[1-\varepsilon, 1+\varepsilon]$, with P_t representing importance sampling. And the clip function is also used to constrain the change in the value function during updates to ensure stability and prevent large policy changes, as shown in Eq. (20), and the error loss function is defined by Eq. (21).

$$P_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{old}(a_t \mid s_t)}, \tag{18}$$

$$J_{PPO}(\theta) = E_t[\min\left(P_t(\theta)A_t, clip\left[P_t(\theta), 1-\varepsilon, 1+\varepsilon\right]A_t\right)], \tag{19}$$

$$v_\phi^{clip}(s_t) = clip\left[v_\phi(s_t), v_{old}(s_t)-\varepsilon, v_{old}(s_t)+\varepsilon\right], \tag{20}$$

$$L_{BL}(\phi) = E_t[\max\left(\left|v_\phi(s_t)-\hat{R}_t\right|, \left|v_\phi^{clip}(s_t)-\hat{R}_t\right|\right)^2], \tag{21}$$

5 Experiments

In order to prove the optimization and computational efficiency of the proposed CT architecture, we compare our CT with AM [9] and FER-AM[11], which are advanced learning based construction methods. We report results for both the greedy and the random versions of our CT model. We conducted comparative trials of the VRPSTW using datasets consisting of 20, 50, and 100 customers, the vehicle capacity values were set to 20, 40, and 50, respectively. For each instance, the customers were randomly generated within a unit square grid [0,1] * [0,1] to ensure uniform distribution of coordinates. Similarly, the time windows were randomly generated within the unit time interval [0,1]. Additionally, the demand of the customer was sampled randomly from the set {0, 1, ..., 30}. The CT proposed in this paper is coded in Python and trained on a server equipped with RTX 3090 and Intel(R) Xeon(R) Platinum 8350C CPU at 2.60GHz.

Significantly, to avoid memory overflow errors, we train 50 epochs for VRPSTW20, 42 epochs for VRPSTW50, 22 epochs for VRPSTW100. However, it can be seen from Table 2 that even with these limitations, our model is still the best performing of the three. With sufficient device memory, our model is highly likely to exhibit even better performance.

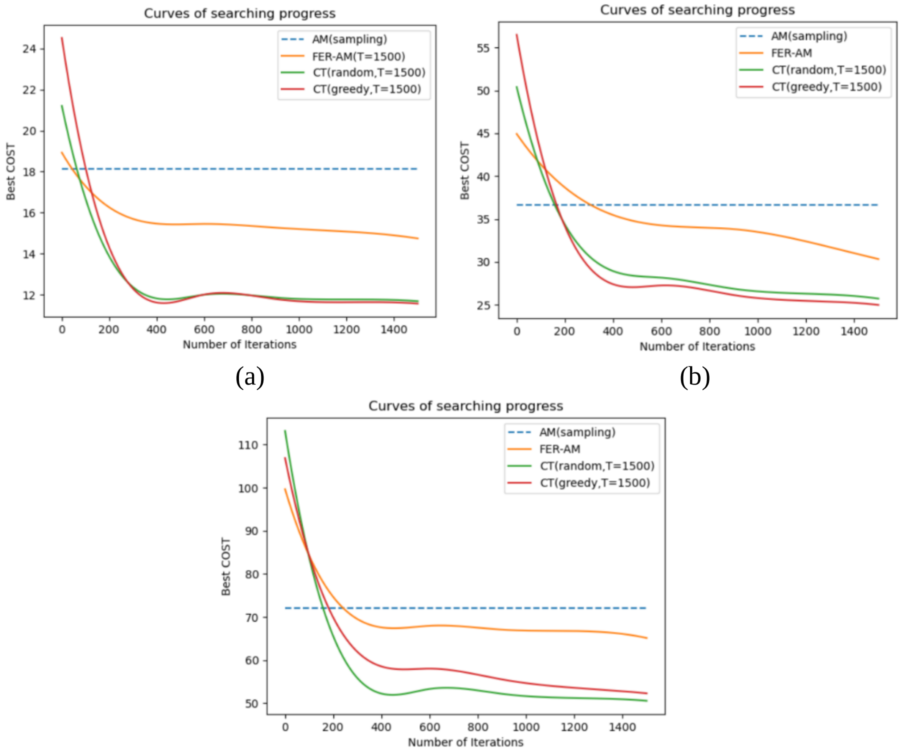
Table 2. Collaborative Transformer (CT) model vs baselines. The average value across all methods.

	n=20		n=50		n=100	
Method	Obj	Gap	Obj	Gap	Obj	Gap
CT(random, T=1500)	13.6809	1.35%	35.2554	5.60%	61.6370	0.63%
CT(random, T=2000)	13.5279	0.21%	33.7399	1.06%	**61.2502**	**0.00%**
CT(greedy, T=1500)	13.7553	1.90%	33.8012	1.24%	62.2698	1.66%
CT(greedy, T=2000)	**13.4991**	**0.00%**	**33.3861**	**0.00%**	62.0093	1.24%
FER-AM(random, T=1500)	17.3620	28.62%	35.6004	6.63%	70.0081	14.30%
AM(sampling)	18.1206	34.24%	36.6300	9.72%	72.0605	17.65%

Table 3. Collaborative Transformer (CT) model vs baselines. The best value across all methods.

	n=20		n=50		n=100	
Method	Obj	Gap	Obj	Gap	Obj	Gap
CT(random, T=1500)	11.6603	1.48%	25.2916	4.51%	50.6554	1.82%
CT(random, T=2000)	11.5266	0.31%	24.6191	1.73%	**49.7515**	**0.00%**
CT(greedy, T=1500)	11.6277	1.48%	24.7685	2.35%	53.9528	8.44%
CT(greedy, T=2000)	**11.4905**	**0.00%**	**24.2002**	**0.00%**	52.1592	4.84%
FER-AM(random, T=1500)	14.7477	28.35%	30.3318	25.34%	65.1173	30.89%
AM(sampling)	18.1206	57.70%	36.6300	51.36%	72.0605	44.84%

We compared the search progress curves of these methods in terms of iteration steps T = 1500, as shown in Fig. 2, where the horizontal coordinate is the iteration step and the vertical coordinate is the average of the best objective values so far for the examples used in Table 3. Since the solution obtained by AM is sampled by the policy network, it does not participate in the iteration step, but in order to have a clearer view of the performance of the CT model, we have likewise indicated the best value of AM(sampling) in the figure with a blue line segment. We can observe that the CT model significantly outperforms AM(sampling) and FER-AM in terms of convergence speed and obtains the best solution in a finite number of steps, which further verifies the effectiveness of our method.



(a) (b)

(c)

Fig. 2. Curves of searching progress for our CT and baselines including AM [9], FER-AM [11]. (a) VRPSTW20. (b) VRPSTW50. (c) VRPSTW100.

6 Conclusions

In this paper, in order to tackle the parcel delivery issues arising in the development of smart cities, we model the scenario as a VRPSTW. To solve this problem, we propose a Collaborative Transformer framework based on deep reinforcement learning architecture for VRPSTW that learns the node information and positional information separately and adjusts the network parameters through the n-step PPO algorithm. This approach mitigates incompatible associations and generates high-quality solutions for the problem. Experimental experiments confirm the superiority of our method against the traditional DRL model. However, due to the nature of feature embeddings, they can only utilize a single objective as the object function, which limits the CT model and cannot be directly applied to multi-objective problems. Therefore, improvements to both the model embedding section and the objective function will be key areas of future research.

References

1. Zhu, Li, et al.: Big data analytics in intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems 20.1, 383-398(2018).
2. Perboli, Guido, and Mariangela Rosano.: Parcel delivery in urban areas: Opportunities and threats for the mix of traditional and green business models. Transportation Research Part C: Emerging Technologies 99, 19-36(2019).
3. Liu, Zhishuo, ngquan Zuo, Mengchu Zhou, Wei Guan, and Yusuf Al-Turki.: Electric vehicle routing problem with variable vehicle speed and soft time windows for perishable product delivery. IEEE Transactions on Intelligent Transportation Systems (2023).
4. Wu, Hongguang, and Yuelin Gao.: An ant colony optimization based on local search for the vehicle routing problem with simultaneous pickup–delivery and time window. Applied Soft Computing 139: 110203 (2023).
5. Li, gwen, Yining Ma, Ruize Gao, Zhiguang Cao, Andrew Lim, Wen Song, and Jie Zhang.: Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. IEEE Transactions on Cybernetics 52, no. 12: 13572-13585(2021).
6. Meng, Ming, and Yun Ma.: Route optimization of electric vehicle considering soft time windows and two ways of power replenishment. Advances in Operations Research 2020: 1-10(2020).
7. Wu, Yaoxin, et al.: Learning improvement heuristics for solving routing problems. IEEE transactions on neural networks and learning systems 33.9 : 5057-5069(2021).
8. Bello I, Pham H, Le Q V, et al.: Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940(2016).
9. Kool, W., Van Hoof, H., & Welling, M.: Attention, learn to solve routing problems!. arxiv preprint arxiv:1803.08475(2018).
10. Nazari, Mohammadreza, et al.: Reinforcement learning for solving the vehicle routing problem. Advances in neural information processing systems 31 (2018).
11. Li J, Ma Y, Cao Z, et al.: Learning feature embedding refiner for solving vehicle routing problems. IEEE Transactions on Neural Networks and Learning Systems (2023).
12. Ma, Yining, et al.: Learning to iteratively solve routing problems with dual-aspect collaborative transformer. Advances in Neural Information Processing Systems 34: 11096-11107(2021).
13. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.: Deep residual learning for image recognition. In IEEE conference on computer vision and pattern recognition, 770–778(2016).
14. Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton.: Layer normalization. ArXiV, Corr: abs/1607.06450(2016).
15. Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin.: Attention is all you need. Advances in neural information processing systems 30 (2017).
16. Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov.: Proximal policy optimization algorithms. arxiv preprint arxiv:1707.06347 (2017).