A Learning-based Capacitated Arc Routing Problem Solver Comparable to Metaheuristics While with Far Less Runtimes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently, neural networks (NN) have made great strides in combinatorial optimization problems (COPs). However, they face challenges in solving the capacitated arc routing problem (CARP) which is to find the minimum-cost tour that covers all required edges on a graph, while within capacity constraints. Actually, NN-based approaches tend to lag behind advanced metaheuristics due to complexities caused by *non-Euclidean graph, traversal direction and capacity constraints.* In this paper, we introduce an NN-based solver tailored for these complexities, which significantly narrows the gap with advanced metaheuristics while with far less runtimes. First, we propose the direction-aware attention model (DaAM) to incorporate directionality into the embedding process, facilitating more effective one-stage decision-making. Second, we design a supervised reinforcement learning scheme that involves supervised pre-training to establish a robust initial policy for subsequent reinforcement fine-tuning. It proves particularly valuable for solving CARP that has a higher complexity than the node routing problems (NRPs). Finally, a path optimization method is introduced to adjust the depot return positions within the path generated by DaAM. Experiments show that DaAM surpasses heuristics and achieves decision quality comparable to state-of-the-art metaheuristics for the first time while maintaining superior efficiency, even in large-scale CARP instances. The code and datasets are provided in the Appendix.

1 Introduction

The capacitated arc routing problem (CARP) [7] is a combinatorial optimization problem, frequently arising in domains such as inspection and search-rescue operations. Theoretically, the CARP is established on an undirected connected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{E}_R)$ that includes a set of nodes $\mathbf{V}$ connected by a set of edges $\mathbf{E}$, and an edge subset $\mathbf{E}_R \subseteq \mathbf{E}$ that needs to be served, called required edges. Each required edge has a demand value which spends the capacity of the vehicle when it is served. In this context, all vehicles start their routes from the depot node $depot \in \mathbf{V}$ and conclude their journey by returning to the same $depot$. The main goal of a CARP solver is to serve all required edges with the lowest total path cost, while ensuring that the vehicle does not exceed its capacity Q.

According to [7], the CARP is recognized as an NP-Hard problem. Among all solver solutions, the Memetic Algorithms (MA) [15, 25], first proposed in 2005, have still maintained unrivaled results in solving the CARP challenge to this day. However, they have struggled with high time costs and the exponential growth of the search space as the problem scale increases. Compared to the traditional methods, NN-based solvers [16, 10, 20] are faster with the assistance of GPU, have therefore gained increasing attention in recent years. However, unlike the decent performance in solving NRP, such as vehicle routing problem (VRP) and travelling salesman problem (TSP), or ARP defined in Euclidean graphs, such as rural postman problem (RPP) and Chinese postman problem (CPP), NN-based methods usually lags far behind the traditional ones in solving CARP. This discrepancy is attributed to the inability of existing methods to effectively reduce the high complexity of solving CARP:

- **Lack of edge direction in embedding learning:** ARP solvers need to determine the edges to be traversed along with the direction of traversal, easy for humans to achieve in one step but extremely challenging for computers. Existing methods didn't encode edge directionality in embedding, making them have to build edge sequences and determine edges' directions separately and leading to path generation without sufficient consideration.
- **Ineffective learning for solving CARP:** CARP is more complex than NRPs and Euclidean ARPs owing to the *non-Euclidean input*, *edge direction*, and *capacity constraints*. Thus, learning methods for NRPs and Euclidean ARPs cannot be directly transferred to solve CARP or work well even if adapted, leaving a lack of effective learning strategies for CARP.

In this paper, we aim to address both above issues and propose an NN-based solver for CARP that competes with the state-of-the-art MA [25] while with far less runtimes. Firstly, we propose the direction-aware attention model (DaAM). It computes embeddings for directed arcs decomposed from undirected edges to align with the nature of ARP, thus avoiding missing direction information and enabling concise one-stage decision-making. Secondly, we design a supervised reinforcement learning method to learn effective heuristics for solving CARP. DaAM is pre-trained to learn an initial policy by minimizing the difference from the decisions made by experts, and fine-tuned on larger-scale CARP instances by Proximal Policy Optimization with self-critical strategy. Finally, to further boost the path quality, we propose a path optimizer (PO) to re-decide the vehicles' optimal return positions by dynamic programming. In the experiments, our method approaches the state-of-the-art MA with an average gap of 5% and is 4% better than the latest heuristics and gains high efficiency.

2 Related Work

2.1 Graph Embedding Learning

Graph embedding [3] aims to map nodes or edges in a graph to a low-dimensional vector space. This process is commonly achieved via graph neural networks (GNNs) [31]. Kipf *et al.* [13] introduced graph convolutional operations to aggregate information from neighboring nodes for updating node representations. Unlike GCN, GAT [27] allowed dynamic node attention during information propagation by attention mechanisms. Other GNN variants [9, 30] exhibited a similar information aggregation pattern but with different computational approaches. In this paper, since an arc is related to the outgoing arc of its endpoint but irrelevant to the incoming arc of that, we use attention mechanisms to capture the intricate relationships between arcs for arc embedding learning.

2.2 Learning for Routing Problems

The routing problem is one of the most classic COPs, and it is mainly categorized into two types according to the decision element: node routing problems and arc routing problems.

Node routing problems (NRPs), such as TSP and VRP, aim to determine the optimal paths that traverse all nodes in the Euclidean space or graphs. As the solutions to these problems are context-dependent sequences of variable size, they cannot be directly modeled by the Seq2Seq model [24]. To address this problem, Vinyals *et al.* [28] proposed the Pointer network (PN) to solving Euclidean TSP, which achieves variable-size output dictionaries by neural attention. Due to the scarcity of labels for supervised learning, Bello *et al.* [2] modeled the TSP as a single-step reinforcement learning problem and trained the PN using policy gradient [29] within Advantage Actor-Critic (A3C) [17] framework. Nazari *et al.* [19] replaced the LSTM encoder in PN with an element-wise projection layer and proposed the first NN-based method to solve the Euclidean VRP and its variants. To better extract correlations between inputs, Kool *et al.* [14] utilized multi-head attention for embedding learning and trained the model using REINFORCE [29] with a greedy baseline. To solve COPs defined on graphs, Khalil *et al.* [11] proposed S2V-DQN to learn heuristics, employing structure2vec [5] for graph embedding learning and n-step DQN [18] for training. While the mentioned NN-based approaches have achieved comparable performance to metaheuristics, they cannot be directly applied to solve ARP due to the modeling differences between ARP and NRP.

Arc routing problems (ARPs) involve determining optimal paths for traversing arcs or edges in graphs, with variants like RPP, CPP, and CARP. Truong *et al.* [26] proposed a DRL framework to address the CPP with load-dependent costs on Euclidean graphs and achieved better solution quality than metaheuristics. However, CARPs are defined on non-Euclidean graphs. Unlike Euclidean graphs with given node coordinates, non-Euclidean graphs require manually extracting and aggregating the node representations, a task that is typically learnable. Although several NN-based algorithms have been proposed, they still lag significantly behind traditional methods. Li *et al.* [16] pioneered the use of the NN-based approach in solving the CARP by transforming it into an NRP. They first determined the sequence of edges and then decided the traversal direction for each edge. Hong *et al.* [10] trained a PN in a supervised manner to select undirected edges in each time step, and also determined the edge traversal direction as post-processing. Ramamoorthy *et al.* [20] proposed to generate an initial tour based on edge embeddings and then split it into routes within capacity constraint. These approaches lack edge directionality encoding, leading to edge selection without sufficient consideration and necessitating a two-stage decision process or an additional splitting procedure.

3 Background

The attention model (AM) [14] exhibits superior effectiveness in solving classic Euclidean COPs due to its attention mechanisms for extracting correlations between inputs. Therefore, we use the AM as the backbone and give a brief review in terms of the TSP. Given an Euclidean graph $\mathbf{G}=(\mathbf{V},\mathbf{E})$, the AM defines a stochastic policy, denoted as $\pi(\boldsymbol{x}|\mathcal{S})$, where $\boldsymbol{x}=(x_0,...,x_{|\mathbf{V}|-1})$ represents a permutation of the node indexes in $\mathbf{V}$, and $\mathcal{S}$ is the problem instance expressing $\mathbf{G}$. The AM is parameterized by $\boldsymbol{\theta}$ as:

$$\pi_\theta(\boldsymbol{x}|\mathcal{S})=\prod\nolimits_{t=1}^{|\mathbf{V}|}\pi_{\boldsymbol{\theta}}(x_t|\mathcal{S},\boldsymbol{x}_{0:t-1}) \tag{1}$$

where t denotes the time step. Specifically, the AM comprises an encoder and a decoder. The encoder first computes initial d_h-dimensional embeddings for each node in $\mathbf{V}$ as h_i^0 through a learned linear projection. It then captures the embeddings of h_i^0 using multiple attention layers, with each comprising a multi-head attention (MHA) sublayer and a node-wise feed-forward (FF) sublayer. Both types of sublayers include a skip connection and batch normalization (BN). Assuming that $l\in\{1,...,N\}$ denotes the attention layer, the l^{th} layer can be formulated as h_i^l:

$$h_i^l=\ \mathrm{BN}^l(\hat{h_i}+\mathrm{FF}^l(\hat{h_i}));\qquad \hat{h_i}=\ \mathrm{BN}^l(h_i^{l-1}+\mathrm{MHA}_i^l(h_0^{l-1},\ldots,h_{|\mathbf{V}|-1}^{l-1})) \tag{2}$$

The decoder aims to append a node to the sequence $\boldsymbol{x}$ at each time step. Specifically, a context embedding $h_{(c)}$ is computed to represent the state at the time step t. Then a single attention head is used to calculate the probabilities for each node based on $h_{(c)}$:

$$u_{(c)j}=\begin{cases}C\cdot\tanh\left(d_h^{-\frac{1}{2}}[\mathbf{W}^Q h_{(c)}]^T\mathbf{W}^K h_j^N\right) & \text{if } j\neq x_{t'}(\forall t'<t)\\ -\infty & \text{otherwise,}\end{cases}$$

$$p_i=\pi_{\boldsymbol{\theta}}(x_t=i|\mathcal{S},\boldsymbol{x}_{0:t-1})=u_{(c)i}/\sum\nolimits_j u_{(c)j} \tag{3}$$

where $\mathbf{W}^Q$ and $\mathbf{W}^K$ are the learnable parameters of the last attention layer. $u_{(c)j}$ is an unnormalized log probability with (c) indicating the context node. C is a constant, and p_i is the probability distribution computed by the softmax function based on $u_{(c)j}$.

4 Method

4.1 Direction-aware Attention Model

In this section, we propose the direction-aware attention model (DaAM). Unlike previous methods that separately learn edge embeddings and determine edge directions, our model encodes direction information directly into the embedding, enabling one-stage decision-making. As shown in Fig. 1, the DaAM makes sequential decisions in two phases to select arcs. **The first phase** is a one-time transformation process, in which the arcs of the input graph are represented as nodes in the new directed complete graph. **The second phase** is executed at each time step, in which GAT is used to aggregate the inter-arc weights. Subsequently, AM is used to select the arc of the next action.

4.1.1 Arc Feature Formulation via Graph Transformation

Graph Transformation Motivated by the need to consider direction when traversing edges, we explicitly encode the edge direction by edge-to-arc decomposition. Let $\mathbf{G}=(\mathbf{V},\mathbf{E},\mathbf{E}_R)$ denotes

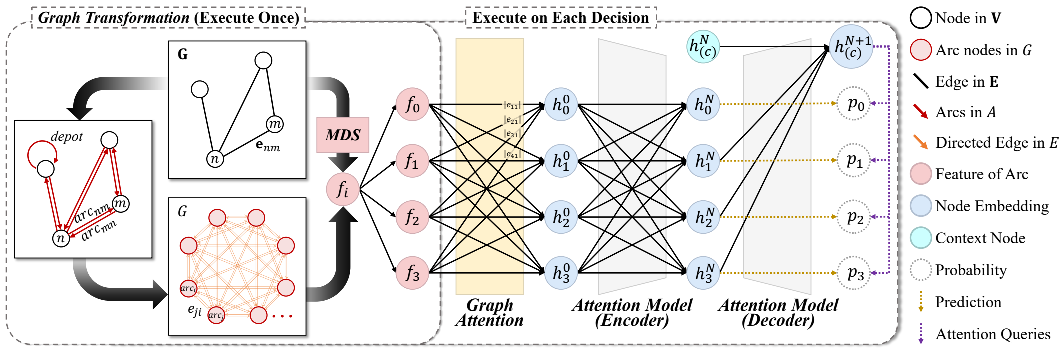

Figure 1: **DaAM Pipeline** consists of two parts. The first part transforms the input graph $\mathbf{G}$ by treating the arcs on $\mathbf{G}$ as nodes of a new directed graph G, only executing once. The second part leverages the GAT and AM to update arc embeddings and select arcs, executing at each time step.

the undirected connected graph as input, where $\mathbf{V}$ is the node set of $\mathbf{G}$, $\mathbf{E}$ is the edge set of $\mathbf{G}$, and $\mathbf{E}_R \subseteq \mathbf{E}$ is the required edge set. Firstly, given that an edge has two potential traversal directions, we decompose each edge $\mathbf{e}_{nm} = (cost_{nm}, demand_{nm}, allow\_serve_{nm}) \in \mathbf{E}_R$ into two arcs $\{arc_{nm}, arc_{mn}\}$ with opposite directions but the same cost, demand and serving state. Here n, m are the indexes of node in $\mathbf{V}$. To simplify the representation below, we replace nm and mn with single-word symbols, such as i and j. In this way of edge decomposition, we obtain a set of arcs denoted as A_R. Secondly, we build a new graph $G = (A_R, E)$. Specifically, each arc in A_R serves as a node in G, and directed edge set E is created, with $e_{ij} \in E$ representing the edge from node arc_i to arc_j. The weight $|e_{ij}|$ represents the total cost of the shortest path from the end node of arc_i to the start node of arc_j. In addition, we treat the depot as a self-loop zero-demand arc that allows for repeated serving, denoted as arc_0. Consequently, we transform the input graph $\mathbf{G}$ into a directed complete graph G. By decomposing all edges in $\mathbf{E}_R$ into arcs, it is natural to directly select the arcs from G during the decision-making. Given that the Floyd-Warshall algorithm is used to calculate the shortest path cost between any pair of nodes in $\mathbf{G}$, the time complexity of our graph transformation is $\max(\mathcal{O}(|\mathbf{E}_R|^2), \mathcal{O}(|\mathbf{V}|^3))$.

Table 1: **Feature Detail** of arc_i at time step t for CARP.

No.	Features	Field	Description	No.	Features	Field	Description
1	is_depot_i	$\mathbb{F}_2$	Is arc_i the depot?	**5**	$allow\_serve_t^{(i)}$	$\mathbb{F}_2$	Is arc_i at time step t allowed to serve?
2	$cost_i$	$\mathbb{R}^+$	Cost of arc_i.	**6**	$mds_{\text{start}(i)}$	$\mathbb{R}^d$	Euclidean coordinates of arc_i's start node.
3	$demand_i$	$\mathbb{R}^+$	Demand of arc_i.	**7**	$mds_{\text{end}(i)}$	$\mathbb{R}^d$	Euclidean coordinates of arc_i's end node.
4	$\lvert e_{x_{t-1}i} \rvert$	$\mathbb{R}^+$	Edge weight from $arc_{x_{t-1}}$ to arc_i.				

Arc Feature Formulation To establish a foundation for decision-making regarding arc selection, the features of the arcs are constructed as input for the subsequent model. Specifically, multi-dimensional scaling (MDS) is used to project the input graph $\mathbf{G}$ into a d-dimensional Euclidean space. The Euclidean coordinates of arc_i's start and end nodes, denoted as $mds_{\text{start}(i)}$ and $mds_{\text{end}(i)}$, are then taken as the features of arc_i to indicate its direction. As shown in Table 1, at time step t, arc_i can be featured as:

$$F_t^{(i)} = (is\_depot_i, cost_i, demand_i, |e_{x_{t-1}i}|, allow\_serve_t^{(i)}, mds_{\text{start}(i)}, mds_{\text{end}(i)}) \tag{4}$$

where x_{t-1} is the index of the selected arc at the last time step, and $t \in [1, +\infty)$. Our feature models arcs rather than edges and encodes the direction attribute of arcs through MDS. Therefore, it is more suitable than previous methods [10, 16] for ARPs that need to consider the direction of traversing.

4.1.2 Arc Relation Encoding via Graph Attention Network

Although AM is efficient in decision-making, according to Eq. (2), it cannot encode the edge weights between nodes in G, an important context feature, during learning. Therefore, we use graph attention network (GAT) [27] to encode such weights. At each time step t, for each arc arc_i, we integrate the weights between arc_i and all arcs in A_R along with their features into the initial embedding of arc_i.

$$c_{ij} = softmax\big(\alpha(\mathbf{W}[\,F_t^{(i)}\,||\,F_t^{(j)}\,||\,|e_{ji}|\,])\big); \qquad h_i^0 = \sigma\big(\sum_{j=0}^{|A_R|-1} c_{ij}\mathbf{W}F_t^{(j)}\big) \tag{5}$$

where $\mathbf{W}$ is a shared learnable parameter, $[\cdot||\cdot]$ is the horizontal concatenation operator, $\alpha(\cdot)$ is a mapping from the input to a scalar, and $\sigma(\cdot)$ denotes the activation function. h_i^0 denotes the initial feature embedding of arc_i, which is taken as the input of subsequent AM. Since G is a complete graph, we use one graph attention layer to avoid over-smoothing [4].

4.1.3 Arc Selection via Attention Model

After aggregating the edge weights of G into the initial embeddings, we utilize AM to learn the final arc embeddings and make arc selection decisions. In the encoding phase described by Eq.2, for each arc $\{arc_i\}$, we leverage N attention layers to process the initial embeddings $\{h_i^0\}$ and obtain the output embeddings of the N^{th} layer, i.e., $\{h_i^N\}$. In the decoding phase, we define the context node applicable to CARP:

$$h_{(c)}^N = \Big[|A_R|^{-1}\sum_{i=0}^{|A_R|-1} h_i^N, h_{x_{t-1}}^N, \delta_t, \Delta_t\Big], t \in [1, +\infty) \tag{6}$$

where x_{t-1} indicates the chosen arc index at time step $t-1$ and x_0 is arc_0. δ_t is the remaining capacity at time step t, $\Delta_t = \Delta(\delta_t > \frac{Q}{2})$ is a variable to indicate whether the vehicle's remaining capacity exceeds half. Finally, according to Eq.(3), the decoder of AM takes the context node $h_{(c)}^N$ and arc embeddings $\{h_i^N\}$ as inputs and calculates the probabilities for all arcs, denoted as p_i. The serviceable arc selected at time step t, i.e., arc_{x_t}, is determined by sampling or greedy decoding.

4.2 Supervised Reinforcement Learning for CARP

The decision-making of selecting arcs can be modeled as a Markov decision process with the following symbols regarding reinforcement learning:

- **State** s_t is the newest path of arcs selected from G: $(arc_{x_0}, ..., arc_{x_{t-1}})$, while the terminal state is s_T with T indicating the final time step.
- **Action** a_t is the selected arc at time step t, i.e., arc_{x_t}. Selecting the action a_t would add arc_{x_t} to the end of the current path s_t and tag the corresponding arcs of arc_{x_t} with their features $allow\_serve$ changed to 0. Notably, arc_0 can be selected repeatedly but not consecutively.
- **Reward** r_t is obtained after taking action a_t at state s_t, which equals the negative shortest path cost from the last arc $arc_{x_{t-1}}$ to the selected arc arc_{x_t}.
- **Stochastic policy** $\pi(a_t|s_t)$ specifies the probability distribution over all actions at state s_t.

We parameterize the stochastic policy of DaAM with θ:

$$\pi(x_t|\,\mathcal{S}, \boldsymbol{x}_{0:t-1}) = \pi_\theta(a_t|s_t) \tag{7}$$

where $\mathcal{S}$ is a CARP instance. Starting from initial state s_0, we get a trajectory $\tau = (s_0, a_0, r_0, ..., r_{T-1}, s_T)$ using π_θ. The goal of learning is to maximize the cumulative reward: $R(\tau) = \sum_{t=0}^{T-1} r_t$. However, due to the high complexity of CARP, vanilla deep reinforcement learning methods learn feasible strategies inefficiently. A natural solution is to minimize the difference between the model's decisions and expert decisions. To achieve this, we employ supervised learning to learn an initial policy based on labeled data and then fine-tune the model through reinforcement learning.

4.2.1 Supervised Pre-training via Multi-class Classification

In the pre-training stage, we consider arc-selection at each time step as a multi-class classification task, and employ the state-of-the-art CARP method MAENS to obtain high-quality paths as the label. Assuming that $y_t \in \mathbb{R}^{|A_R|}$ denotes the one-hot label vector at time step t of any path, with $y_t^{(k)}$ indicating each element. We utilize the cross-entropy loss to train the policy represented in Eq. (7):

$$L = -\sum_{t=0}^{T-1}\sum_{k=0}^{|A_R|-1} y_t^{(k)} \log\big(\pi_\theta(arc_k|s_t)\big) \tag{8}$$

We use the policy optimized by cross-entropy, denoted as π_s, to initialize the policy network π_θ and as the baseline policy π_b in reinforcement learning.

4.2.2 Reinforcement Fine-tuning via PPO with self-critical strategy

During the fine-tuning phase, we use Proximal Policy Optimization (PPO) to optimize our model $\pi_\theta(a_t|s_t)$ due to its outstanding stability in policy updates. Considering the low sample efficiency in reinforcement learning, we employ a training approach similar to self-critical training [21] to reduce gradient variance and expedite convergence. Specifically, We use another policy π_b to generate a trajectory and calculate its cumulative reward, serving as a baseline function. Our optimization objective is based on PPO-Clip [23]:

$$\mathbb{E}_{(s,a)\sim\pi_b}\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_b(a|s)}\big(R(\tau_s^\theta)-R(\tau_s^b)\big),\text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_b(a|s)},1-\epsilon,1+\epsilon\right)\big(R(\tau_s^\theta)-R(\tau_s^b)\big)\right)\right] \quad (9)$$

where s is used to replace current state s_t for symbol simplification, and a for a_t. $\text{clip}(w, v_{\min}, v_{\max})$ denotes constraining w within the range $[v_{\min}, v_{\max}]$, and ϵ is a hyper-parameter. τ_s^θ denotes a trajectory sampled by π_θ with s as the initial state, while τ_s^b for the trajectory greedily decoded by π_b. In greedy decoding, the action with the maximum probability is selected at each step. $R(\tau_s^\theta) - R(\tau_s^b)$ serves as an advantage measure, quantifying the advantage of the current policy π_θ compared to π_b. We maximize Eq. (9) through gradient descent, which forces the model to select actions that yield higher advantages. The baseline policy's parameters are updated if π_θ outperforms π_b.

4.3 Path Optimization via Dynamic Programming

The complexity of the problem is heightened by the increasing capacity constraint, making it challenging for the neural network to make accurate decisions regarding the depot return positions. In this section, we propose a dynamic programming (DP) based strategy to assist our model in optimizing these positions. Assuming that $\boldsymbol{P}$ is assigned with the terminal state $s_T = (arc_{x_0}, arc_{x_1}, ..., arc_{x_{T-1}})$, representing a generated path. Initially, we remove all the depot arcs in $\boldsymbol{P}$ to obtain a new path $\boldsymbol{P}^{'} = (arc_{x_0'}, arc_{x_1'}, ..., arc_{x_{T'-1}'})$, where $\{x_i' | i \in [0, T'-1]\}$ denotes a subsequence of $\{x_i | i \in [0, T-1]\}$. Subsequently, we aim to insert several new depot arcs into the path $\boldsymbol{P}^{'}$ to achieve a lower cost while adhering to capacity constraints. To be specific, we recursively find the return point that minimizes the overall increasing cost, which is implemented by the state transition equation as follows:

$$f(\boldsymbol{P}^{'}) = \min_i (f(\boldsymbol{P}^{'}_{0:i}) + SC(arc_{x_i'}, arc_0) + SC(arc_0, arc_{x_{i+1}'}) - SC(arc_{x_i'}, arc_{x_{i+1}'}))$$
$$\text{s.t.} \quad 0 \leq \ i < T^{'} - 1, \quad \sum_{j=i+1}^{T^{'}-1} demand_{x_j'} \leq Q \quad (10)$$

where $SC(arc_{x_i'}, arc_0) = |e_{x_i'0}|$ denotes the shortest path cost from $arc_{x_i'}$ to the depot. Q is the vehicle capacity. According to Eq. (10), we insert the depot arc arc_0 after an appropriate position $arc_{x_i'}$, which meets with the capacity constraint of the subpath $\boldsymbol{P}^{'}_{i+1:T'-1}$. $f(\cdot)$ denotes a state featuring dynamic programming. By enumerating the position i, we compute the minimum increasing cost $f(\boldsymbol{P}^{'})$ utilizing its sub-state $f(\boldsymbol{P}^{'}_{0:i})$. The final minimum cost for path $\boldsymbol{P}$ is $f(\boldsymbol{P}') + g(\boldsymbol{P}')$, here $g(\boldsymbol{P}')$ is the unoptimized cost of $\boldsymbol{P}'$. Since $\boldsymbol{P}^{'}$ includes only the required edges, i.e., $T' = |\mathbf{E}_R|$, the time complexity of DP is $\mathcal{O}(|\mathbf{E}_R|^2)$. During Path Optimization, we use beam search to generate two paths with the trained policy, one with capacity-constrained and one without. Both paths are optimized using DP and the one with the minimum cost is selected as the final result.

5 Experiments

5.1 Setup

Problem Instances. We extracted sub-graphs from the roadmap of Beijing, China, obtained from OpenStreetMap [8], to create CARP instances for both the training and testing phases. All instances are divided into seven datasets, each representing different problem scales, as presented in Table 2. Each dataset consists of 30,000 instances, further divided into two **disjoint** subsets: 20,000 instances for training and the remaining for testing. For each instance, the vehicle capacity is set to 100.

Implementation Details. Our neural network is implemented using the PyTorch framework and trained on a single NVIDIA RTX 3090 GPU. The heuristics and metaheuristics algorithms are

Table 2: **Datasets information**. $|\mathbf{V}|$ is the number of nodes, $|\mathbf{E}_R|$ is the number of required edges. $demand$ represents the demand range for each required edge. Each dataset has 20,000 training instances and 10,000 test instances.

CARP instances	$\|\mathbf{V}\|$	$\|\mathbf{E}_R\|$	$demand$	**CARP instances**	$\|\mathbf{V}\|$	$\|\mathbf{E}_R\|$	$demand$
Task 20	25-30	20	5-10	Task 200	205-210	200	1000
Task 40	45-50	40	5-10	Task 300	305-310	300	1000
Task 60	65-70	60	5-10	Task 400	405-410	400	1000
Task 80	85-90	80	5-10	Task 500	505-510	500	1000
Task 100	105-110	100	5-10	Task 600	605-610	600	1000

Table 3: **Solution quality comparison**. All methods are evaluated on 10,000 CARP instances in each scale. We measure the gap (%) between different methods and MAENS. Methods marked with an asterisk were originally proposed for NRP, but we modified them to solve CARP. The best results are indicated in bold, while the second-best results are underlined.

Method	Task20		Task40		Task60		Task80		Task100	
	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)
MAENS [25]	474	0	950	0	1529	0	2113	0	2757	0
PS [6]	544	14.72	1079	13.56	1879	22.84	2504	18.49	3361	21.90
PS-Ellipse [22]	519	9.49	1006	5.89	1709	11.77	2299	8.80	3095	12.26
PS-Efficiency [1]	514	8.44	1007	<u>6.00</u>	1684	10.14	2282	8.00	3056	10.85
PS-Alt1 [1]	514	8.44	1007	<u>6.00</u>	1685	10.20	2283	8.04	3057	10.88
PS-Alt2 [1]	521	9.92	1009	6.21	1720	12.49	2314	9.51	3102	12.51
S2V-DQN* [11]	590	24.42	1197	26.02	1900	24.23	2820	33.43	3404	23.42
VRP-DL* [19]	528	11.39	1193	25.57	2033	32.96	2898	37.15	3867	40.26
DaAM (SL)	509	7.43	1066	12.24	-	-	-	-	-	-
DaAM (SL+RL)	495	<u>4.48</u>	1009	6.19	1639	<u>7.16</u>	2275	<u>7.67</u>	2980	<u>8.06</u>
DaAM (SL+RL+PO)	482	**1.65**	992	**4.39**	1621	**5.98**	2255	**6.70**	2958	**7.28**

evaluated on an Intel Core i9-7920X with 24 cores and a CPU frequency of 4.4GHz. We optimize the model using Adam optimizer [12]. The dimension of MDS coordinates d is set to 8, and the learning rate is set to $1e^{-4}$. We set ϵ in the PPO training at 0.1. Notably, our PPO training does not incorporate discounted cumulative rewards, i.e., γ is set to 1.

Metrics and Settings. For each method and dataset, We compute the mean tour cost across all test instances, indicated by "Cost". Employing the state-of-the-art MAENS [25] as a baseline, we measure the "Cost" gap between alternative algorithms and MAENS, indicated by "Gap". We compare our method against the heuristic Path-Scanning algorithms (PS) [6, 22, 1] and two NN-based algorithms. In the absence of publicly available code for prior NN-based CARP methods, we modify two NN-based NRP solvers to suit CARP, i.e, S2V-DQN [11] and VRP-DL [19]. Note that, for S2V-DQN, we replace structure2vec with GAT to achieve more effective graph embedding learning. For our method, we incrementally add supervised pre-training (SL), reinforcement learning fine-tuning (RL), and path optimization (PO) to assess the effectiveness of our training scheme and optimization, respectively. Due to the excessively long computation times of MAENS on larger-scale datasets, SL is only performed on Task20, Task 30, and Task40. The batch size for SL is set to 128. During the RL stage, greedy decoding is used to generate solutions, and except for the Task20 dataset, we utilize the training results obtained from the preceding smaller-scale dataset to initialize the model. The beam width in the PO stage is set to 2. For each dataset, we compare the mean cost of different methods on 10,000 problem instances.

5.2 Evaluation Results

Solution Quality Table 6 shows the result. Our algorithm outperforms all heuristic and NN-based methods across all scales, achieving costs comparable to MAENS, trailing by less than 8%. The advantage over PS demonstrates that neural networks can learn more effective policies than hand-crafted ones, attributed to our well-designed modeling approach. Moreover, as the problem scale increases, it becomes time-consuming to obtain CARP annotation by MAENS. Therefore, we leverage the model pre-trained on small-scale instances as the initial policy for RL fine-tuning on Task50, Task60, Task80, and Task100, yielding commendable performance. This proves the generalization of our training scheme across varying problem scales. The performance gap with MAENS highlights our algorithm's superiority in CARP-solving approaches.

Table 4: **Generalization to larger problem instances**. All methods are evaluated on 10,000 CARP instances in each scale. For DaAM, we employ the policy trained on Task100. The best results are indicated in bold, while the second-best results are underlined.

Method	Task200 Cost	Task300 Cost	Task400 Cost	Task500 Cost	Task600 Cost
PS-Ellipse [22]	4240	6563	8600	10909	13377
PS-Efficiency [1]	4233	6544	8583	<u>10883</u>	<u>13338</u>
PS-Alt1 [1]	4233	6544	<u>8580</u>	10884	13338
PS-Alt2 [1]	4244	6569	8606	10922	13393
DaAM (SL+RL)	<u>4189</u>	<u>6372</u>	8610	10938	13340
DaAM (SL+RL+PO)	**4132**	**6281**	**8473**	**10633**	**13100**

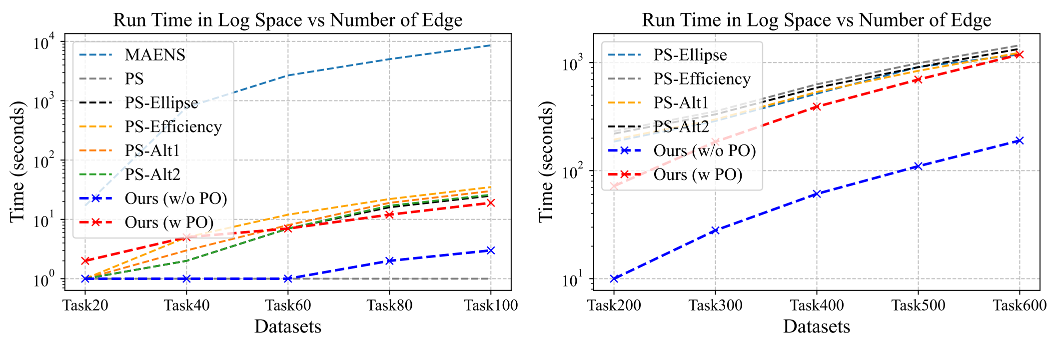

Figure 2: **Run time comparison**. For each dataset, the total run time of each method on 100 CARP instances is shown.

Generalization Ability. In Table 4, we assess DaAM's generalization on large-scale CARP instances using the policy trained on Task100. We remove MAENS and PS due to failing to run on large-scale graphs, and remove S2V-DQN and VRP-DL due to poor performance. Although DaAM is not trained on large-scale instances, it achieves or even exceeds the performance of PS, which shows its potential application on larger-scale CARP instances.

Run Time. We compare the total time required for solving 100 CARP instances across datasets Task20 to Task100 datasets using our method, MAENS, and PS algorithms, and show the run time in log space. For datasets Task200 to Task600, we compare the same metric using variants of PS and out method. For our method, we measured the solving time with and without PO. Fig. 2 demonstrates that our method exhibits a significant speed advantage over MAENS, even outperforming variants of PS [1] on most datasets. In comparison, the consumption time of MAENS increases exponentially as the problem scale increases. Our method efficiently generates paths for large-scale CARP instances by leveraging GPU data-level parallelism and CPU instruction-level parallelism.

Table 5: **Costs of DaAM** using different encoder.

Method	Task30	Task40	Task50	Task60
MDS	743	1017	1338	1699
GAT	746	1019	**1317**	1684
MDS + GAT	**741**	**1011**	1322	**1683**

Effectiveness of Combining MDS and GAT. To evaluate the combination of MDS and GAT for embedding exhibiting, we individually evaluate the performance of models using only MDS or GAT, as well as their combined performance. The experiment is conducted on Task30, Task40, Task50, and Task60 by comparing the average performance of 1,000 instances on each dataset. In the RL stage, we use the policy pre-trained on Task30 for initialization. Table 5 indicates that using MDS or GAT individually yields worse quality in most cases, highlighting that combining MDS and GAT enhances the model's capacity to capture arc correlations. Fig. 3 depicts the convergence trends in these scenes, which shows that the synergy of MDS and GAT contributes to the stability of training.

Solution Visualization. For a more intuitive understanding of the paths generated by different methods, we visualize and compare the results of our method with PS [6] and MAENS across four road scenes in Beijing. Fig. 4 visualizes all results alongside scene information. We observe that our model obtains similar paths with MAENS since we leverage the annotation generated by MAENS for

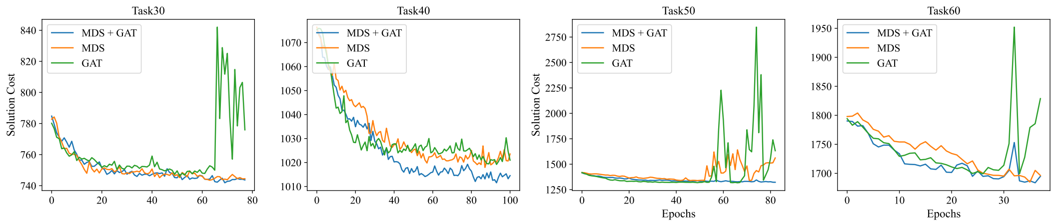

Figure 3: **Convergence trends** of different methods in reinforcement learning training.

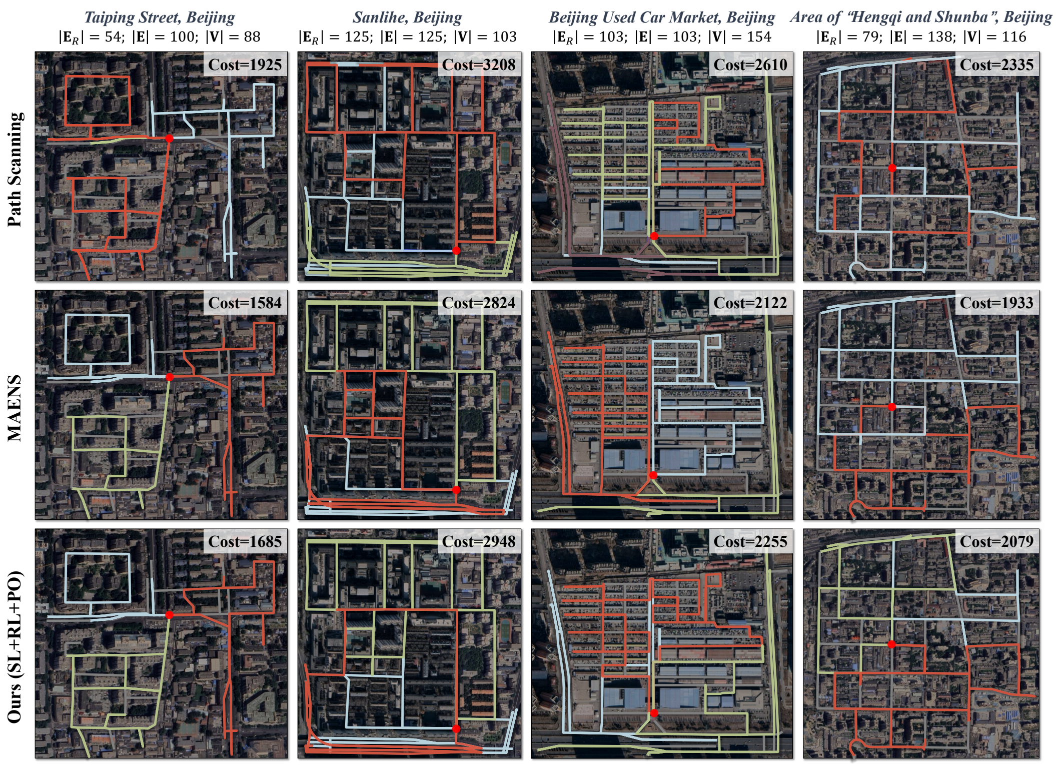

Figure 4: **Qualitative comparison** in four real street scenes. The paths are marked in different colors, with gray indicating roads that do not require service and red points indicating depots.

supervised learning. MAENS paths exhibit superior spatial locality, clearly dividing the scene into regions, whereas PS paths appear more random.

6 Conclusion and Limitations

In this paper, we propose a learning-based CARP solver that competes with state-of-the-art metaheuristics. Firstly, we encode the potential serving direction of edges into embeddings, ensuring that edge directionality is taken into account in decision-making. Secondly, we present a supervised reinforcement learning approach that effectively learns policies to solve CARP. With the aid of these contributions, our method surpasses all heuristics and achieves performance comparable to metaheuristics for the first time while maintaining excellent efficiency.

Limitations and future work. Decomposing undirected edges increases the decision elements, which complicates the problem and may widens the gap between DaAM and traditional state-of-the-art approaches as the problem instance scale increases. Our future work focuses on designing an efficient graph transformation method that does not significantly increase problem complexity.

References

[1] Rafael Kendy Arakaki and Fabio Luiz Usberti. An efficiency-based path-scanning heuristic for the capacitated arc routing problem. *Computers & Operations Research (COR)*, 103:288–295, 2019.

[2] Irwan Bello*, Hieu Pham*, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[3] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering (TKDE)*, 30(9):1616–1637, 2018.

[4] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI conference on artificial intelligence (AAAI)*, 2020.

[5] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning (ICML)*, 2016.

[6] Bruce L Golden, James S DeArmon, and Edward K Baker. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research (COR)*, 10(1):47–59, 1983.

[7] Bruce L Golden and Richard T Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

[8] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 7(4):12–18, 2008.

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems (NeurIPS)*, 2017.

[10] Wenjing Hong and Tonglin Liu. Faster capacitated arc routing: A sequence-to-sequence approach. *IEEE Access*, 10:4777–4785, 2022.

[11] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in neural information processing systems (NeurIPS)*, 2017.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[13] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[14] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations (ICLR)*, 2019.

[15] Natalio Krasnogor and James Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE transactions on Evolutionary Computation (TEVC)*, 9(5):474–488, 2005.

[16] Han Li and Guiying Li. Learning to solve capacitated arc routing problems by policy gradient. In *IEEE Congress on Evolutionary Computation (CEC)*, 2019.

[17] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning (ICML)*, pages 1928–1937, 2016.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[19] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In *Advances in neural information processing systems (NeurIPS)*, 2018.

[20] Muhilan Ramamoorthy and Violet R. Syrotiuk. Learning heuristics for arc routing problems. *Intelligent Systems with Applications (ISWA)*, 21:200300, 2024.

[21] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

[22] Luís Santos, João Coutinho-Rodrigues, and John R Current. An improved heuristic for the capacitated arc routing problem. *Computers & Operations Research (COR)*, 36(9):2632–2637, 2009.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[24] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NeurIPS)*, 2014.

[25] Ke Tang, Yi Mei, and Xin Yao. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation (TEVC)*, 13(5):1151–1166, 2009.

[26] Cong Dao Tran and Truong Son Hy. Graph attention-based deep reinforcement learning for solving the chinese postman problem with load-dependent costs. *arXiv preprint arXiv:2310.15516*, 2023.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[28] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in neural information processing systems (NeurIPS)*, 2015.

[29] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[30] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning (ICML)*, 2019.

[31] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems (TNNLS)*, 32(1):4–24, 2020.

A Appendix

A.1 Source Code and Dataset

The source code of DaAM and the datasets used for testing are available at DaAM. Once the paper is accepted, we will promptly release the source code and datasets.

A.2 Pseudocode of PPO with self-critical strategy

Algorithm 1 presents the pseudocode for the PPO training algorithm we used. In the code implementation, the trajectory τ_s^θ can be replaced by (s, a)'s original trajectory τ_o for efficiency. Once τ_o is sampled, the cumulative rewards from any state $s \in \tau_o$ can be quickly computed.

Algorithm 1 PPO algorithm with self-critical strategy

Input: batch size B, number of episodes K, train instances $\mathcal{P}$, test instances $\mathcal{T}$
Initialize policies $\pi_\theta, \pi_b \leftarrow \pi_s$

for episode $k = 1$ to K **do**
  Initialize data batch $\mathcal{M}, \mathcal{M}' \leftarrow ()$
  while $|\mathcal{M}| < B$ **do**
    Sample a CARP instance $\mathcal{S}$ from $\mathcal{P}$
    Sample $\tau_o = (s_0, a_0, \ldots, s_T)$ from $\mathcal{S}$ using π_b
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{(s_0, a_0), \ldots, (s_{T-1}, a_{T-1})\}$
  end while
  for each $(s, a) \in \mathcal{M}$ **do**
    Generate trajectory τ_s^θ using π_θ from s by sampling
    Generate trajectory τ_s^b using π_b from s by greedy decoding
    Compute advantage $\mathcal{A}_s = R(\tau_s^\theta) - R(\tau_s^b)$
    $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{(s, a, \mathcal{A}_s)\}$
  end for
  Update π_θ using Adam over (9) based on $\mathcal{M}'$
  if π_θ outperforms π_b on $\mathcal{T}$ **then**
    $\pi_b \leftarrow \pi_\theta$
  end if
end for

A.3 Experimental Results of Additional Datasets

For small-scale problem instances, we generated two additional datasets, Task30 and Task50. In Task30 the range of $|V|$ is 25-30, while in Task50, it spans 55-60. Correspondingly, $|\mathbf{E_R}|$ is set to 30 and 50, respectively The demand for each edge ranges from 5 to 10 in both tasks. Table 6 is the complete experimental data from the solution quality experiments.

Table 6: **Solution quality comparison**. All methods are evaluated on 10,000 CARP instances in each scale. We measure the gap (%) between different methods and MAENS. Methods marked with an asterisk were originally proposed for NRP, but we modified them to solve CARP. The gray indicates that MAENS is taken as the baseline when calculating "Gap". The best results are indicated in bold, while the second-best results are underlined.

Method	Task20		Task30		Task40		Task50		Task60		Task80		Task100	
	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)
MAENS [25]	474	0.00	706	0.00	950	0.00	1222	0.00	1529	0.00	2113	0.00	2757	0.00
PS [6]	544	14.72	859	21.76	1079	13.56	1448	18.45	1879	22.84	2504	18.49	3361	21.90
PS-Ellipse [22]	519	9.49	798	13.03	1006	5.89	1328	8.67	1709	11.77	2299	8.80	3095	12.26
PS-Efficiency [1]	514	8.44	790	11.90	1007	<u>6.00</u>	1311	7.28	1684	10.14	2282	8.00	3056	10.85
PS-Alt1 [1]	514	8.44	791	12.04	1007	<u>6.00</u>	1312	7.36	1685	10.20	2283	8.04	3057	10.88
PS-Alt2 [1]	521	9.92	802	13.60	1009	6.21	1336	9.33	1720	12.49	2314	9.51	3102	12.51
S2V-DQN* [11]	590	24.42	880	24.65	1197	26.02	1520	24.32	1900	24.23	2820	33.43	3404	23.42
VRP-DL* [19]	528	11.39	848	20.11	1193	25.57	1587	29.87	2033	32.96	2898	37.15	3867	40.26
DaAM (SL)	509	7.43	785	11.18	1066	12.24	-	-	-	-	-	-	-	-
DaAM (SL+RL)	495	<u>4.48</u>	741	<u>5.05</u>	1009	6.19	1303	<u>6.58</u>	1639	<u>7.16</u>	2275	<u>7.67</u>	2980	<u>8.06</u>
DaAM (SL+RL+PO)	**482**	**1.65**	**725**	**2.73**	**992**	**4.39**	**1283**	**5.07**	**1621**	**5.98**	**2255**	**6.70**	**2958**	**7.28**

A.4 Licences of Assets Used for Experiments

The code we used does not require special consent from the authors. We follow their licenses as specified below:

- https://github.com/wouterkool/attention-learn-to-route: MIT Licence.
- https://github.com/Hanjun-Dai/graph_comb_opt: MIT Licence.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Abstract and Introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See section **Conclusion and Limitations**.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This papar does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: This papar discusses the detail to reproduce the main experimental results of the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The source code and datasets are provided in the Appendix.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See section **Experiments**.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Since the experimental results are deterministic, we did not repeat the experiments multiple times. However, to reduce errors, we calculated the average over 10,000 problem instances for each dataset of any scale.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See section **Experiments**.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The aim of our work is to provide a better solution for a class of combinatorial optimization problems, though it is difficult to predict its impact on society.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See the **Appendix**.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: See the **Appendix**.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.