Prompt Learning for Generalized Vehicle Routing

Fei Liu[1], **Xi Lin**[1], **Weiduo Liao**[1], **Zhenkun Wang**[2,*], **Qingfu Zhang**[1,*], **Xialiang Tong**[3], and **Mingxuan Yuan**[3]
[1]City University of Hong Kong
[2]Southern University of Science and Technology
[3]Huawei Noah's Ark Lab
{fliu36-c,xi.lin,weiduliao2-c}@my.cityu.edu.hk, wangzk3@sustech.edu.cn, qingfu.zhang@cityu.edu.hk, {tongxialiang,yuan.mingxuan}@huawei.com

Abstract

Neural combinatorial optimization (NCO) is a promising learning-based approach to solving various vehicle routing problems without much manual algorithm design. However, the current NCO methods mainly focus on the in-distribution performance, while the real-world problem instances usually come from different distributions. A costly fine-tuning approach or generalized model retraining from scratch could be needed to tackle the out-of-distribution instances. Unlike the existing methods, this work investigates an efficient prompt learning approach in NCO for cross-distribution adaptation. To be concrete, we propose a novel prompt learning method to facilitate fast zero-shot adaptation of a pre-trained model to solve routing problem instances from different distributions. The proposed model learns a set of prompts among various distributions and then selects the best-matched one to prompt a pre-trained attention model for each problem instance. Extensive experiments show that the proposed prompt learning approach facilitates the fast adaptation of pre-trained routing models. It also outperforms existing generalized models on both in-distribution prediction and zero-shot generalization to a diverse set of new tasks. Our code implementation is available online at https://github.com/FeiLiu36/PromptVRP.

1 Introduction

The Vehicle Routing Problem (VRP) can be found in many real-world applications such as logistics, transportation, retail distribution, waste collection, and manufacturing [Toth and Vigo, 2014]. Its objective is to manage a fleet of vehicles optimally, minimizing the total cost while satisfying the demands of customers. As an NP-hard problem, exact methods can hardly solve it efficiently, while heuristic algorithms require costly handcrafted designs with domain knowledge. In contrast, neural combinatorial optimization (NCO), which learns a heuristic based on neural networks, has received growing attention [Bengio *et al.*, 2021; Raza *et al.*, 2022; Bai *et al.*, 2023; Bogyrbayeva *et al.*, 2024] due to its potential ability to generate high-quality solutions without much human effort [Vinyals *et al.*, 2015; Kool *et al.*, 2018; Bogyrbayeva *et al.*, 2024].

Most existing neural combinatorial optimization methods focus on solving in-distribution instances, while real-world routing problem instances are typically from diverse distributions. Therefore, their performance could deteriorate dramatically on out-of-distribution instances [Bi *et al.*, 2022; Zhou *et al.*, 2023]. Recent efforts have focused on enhancing the generalization capabilities for out-of-distribution tasks [Jiang *et al.*, 2022; Bi *et al.*, 2022; Fu *et al.*, 2021; Pan *et al.*, 2023; Manchanda *et al.*, 2023; Drakulic *et al.*, 2023; Jiang *et al.*, 2023; Zhou *et al.*, 2023]. The majority of these approaches involve training a single generalized model using meta-learning techniques [Jiang *et al.*, 2022; Bi *et al.*, 2022; Manchanda *et al.*, 2023; Zhou *et al.*, 2023], which can be adapted effectively to tackle instances from different distributions. However, these methods often necessitate complex and time-intensive meta-learning-based training, while the learning capacity is constrained by the fixed model structure.

This paper proposes a novel approach, which uses prompt learning [Zhou *et al.*, 2022; Liu *et al.*, 2023] to enable fast zero-shot adaptation of a pre-trained NCO model to tackle out-of-distribution routing problem instances. As shown in Figure 1, we keep the pre-trained encoder-decoder NCO model fixed and efficiently learn a pool of key-prompt pairs incorporated into the model for handling different problem instances from diverse distributions. The cross-distribution information is learned through the shared prompts. For solving a new problem instance, the most suitable key will be automatically selected, and its matched prompt will be used to adjust the pre-trained NCO model in a zero-shot manner for better inference. In this way, the proposed prompt learning method can efficiently extend the learning capacity of the pre-trained NCO model, demonstrating competitive generalization performance.

The contributions of this work are summarized as follows:

- We investigate how to incorporate prompt learning into neural combinatorial optimization and propose the first prompt learning method for solving cross-distribution vehicle routing problems.
- We develop a novel and efficient prompt-based attention

[*]corresponding author

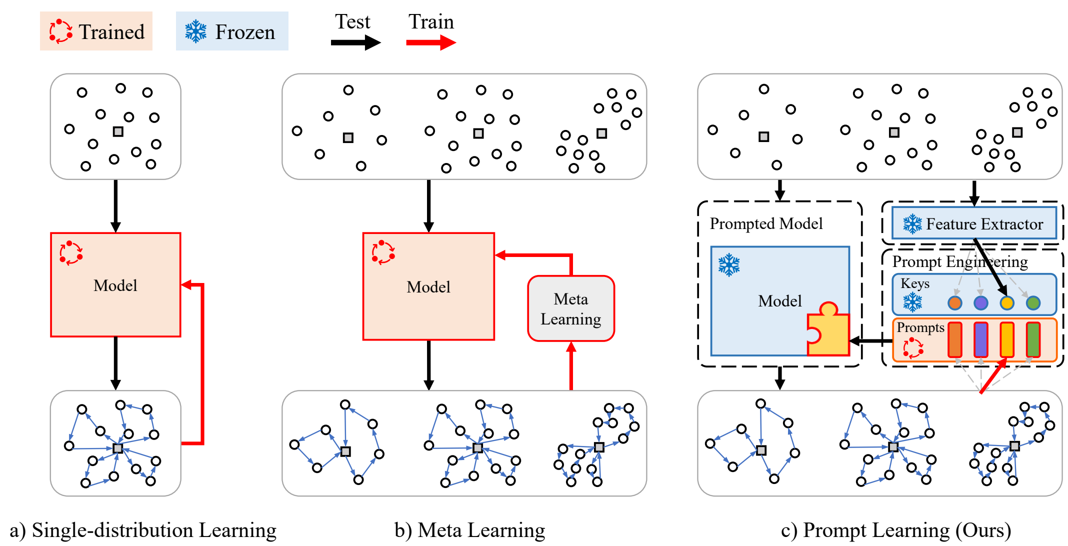

Figure 1: Three approaches for cross-distribution neural combinatorial optimization. **a) Single-distribution Learning:** Single-distribution learning focuses on solving problem instances from the same distribution, and hence its performance usually significantly deteriorates for out-of-distribution cases. **b) Meta Learning:** Meta learning builds a single model to handle problem instances from diverse distributions. It requires a complicated and time-consuming training strategy, while the learning capacity might be limited by the static model structure. **c) Prompt Learning (Ours):** The proposed prompt learning incorporates a trainable key-prompt pool into a frozen NCO model to tackle different problem instances across diverse distributions. For inference, it can automatically select the most suitable prompt for a given instance, and adjust the prompt-based attention in a zero-shot manner to obtain better performance.

model to tackle different routing problem instances from diverse distributions via fast zero-shot adaption.

- We evaluate our proposed prompt learning method on extensive cross-distribution routing instances as well as benchmark instances. With a much lower training cost, our method achieves superior performance compared to existing meta learning methods.

2 Related Works

2.1 Neural Combinatorial Optimization (NCO)

NCO intends to automatically learn a heuristic based on neural networks for solving combinatorial optimization problems. Compared to the other approaches, such as exact methods and heuristic algorithms, it usually generate high-quality solutions with a fast runtime [Bengio *et al.*, 2021]. As a result, NCO has gained much attention in the past decade [Bengio *et al.*, 2021; Bogyrbayeva *et al.*, 2024]. As one of the most important combinatorial optimization problems, the vehicle routing problems have been extensively studied in NCO works [Vinyals *et al.*, 2015; Bello *et al.*, 2016; Nazari *et al.*, 2018; Kool *et al.*, 2018; Li *et al.*, 2022].

There are mainly two categories of works along this line: the end-to-end construction-based methods [Vinyals *et al.*, 2015; Bello *et al.*, 2016; Kool *et al.*, 2018; Kwon *et al.*, 2020; Joshi *et al.*, 2022] and the improvement-based methods [Chen and Tian, 2019; Hottung and Tierney, 2019; Chen and Tian, 2019; Kool *et al.*, 2022]. The former aims to construct a solution without any assistance from classical solvers, while the latter incorporates additional algorithms to improve performance. This work focuses on the construction-based method.

2.2 NCO for Cross-distribution Routing Problem

Several meta learning methods have been developed to improve the out-of-distribution generalization performance for routing problems. Jiang [2022] and Bi [2022] explored the robust optimization over multiple geometrical distributions. Several works [Fu *et al.*, 2021; Pan *et al.*, 2023; Manchanda *et al.*, 2023; Drakulic *et al.*, 2023] studied the generalization to large-scale problems. Zhou [2023] considered generalization in terms of both problem size and geometrical distribution. Most of the existing works adopt a single generalized model and use meta learning methods to improve cross-distribution performance, which might lead to time-consuming training and constrained learning capacity.

2.3 Prompt Learning

Prompt learning has recently gained significant attention in many research areas, such as natural language processing (NLP) [Liu *et al.*, 2023], computer vision (CV) [Jia *et al.*, 2022; Zhou *et al.*, 2022; Ge *et al.*, 2023], and reinforcement learning (RL) [Xu *et al.*, 2022]. In NLP, seminal works like GPT-3 [Brown *et al.*, 2020] and InstructGPT [Ouyang *et al.*, 2022] showcase the effectiveness of prompts in guiding text generation for diverse tasks. In CV, prompt learning can enable few-shot learning [Zhang *et al.*, 2023] and improves image captioning [Wang *et al.*, 2023] by conditioning on specific instructions. In RL, prompt learning can leverage the flexible adaption of prompts to enhance the few-shot policy generalization performance [Xu *et al.*, 2022].

In recent years, many well-trained models have been developed for combinatorial optimization [Kool *et al.*, 2018; Kwon *et al.*, 2020; Bogyrbayeva *et al.*, 2024]. However, the effective utilization of these pre-trained models has not been thoroughly investigated. This paper proposes a prompt learning method to efficiently adapt a fixed pre-trained model for addressing cross-distribution vehicle routing problems.

3 Prompt Learning for Routing

3.1 Problem Formulation

We denote a basic capacited vehicle routing problem (CVRP) on an undirected graph $G = (V, E)$. $V = \{v_0, \ldots, v_n\}$, where v_0 is the depot and $v_1, \ldots, v_n$ are the n customers. $V_c = \{v_1, \ldots, v_n\}$ is the customer set. For the i-th customer, there is a demand d_i. $E = \{e_{ij}\}, i, j \in \{1, \ldots, n\}$ are the edges between every two nodes. For each edge e_{ij}, there is an associated cost (distance) c_{ij}. A fleet of homogeneous vehicles with a capacity of C is sent out from the depot to visit the customers and return to the depot. All the customer's demands should be served. Each customer must be visited once. The objective is to minimize the total traveling distance of all the routes with all the constraints satisfied.

3.2 Main Idea and Basic Framework

The typical constructive-based NCO methods [Kool *et al.*, 2018; Kwon *et al.*, 2020] use an attention-based encoder-decoder model to directly construct a valid solution (e.g., a tour) for the mentioned routing problem. They learn the best model parameters for the attention model to minimize the total distances. In this case, the objective of model training would be:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathcal{G} \sim p(\mathcal{G})} \mathcal{L}\left(\tau \mid \theta, \mathcal{G}\right) \tag{1}$$

where $\mathcal{G}$ represents the given instance, θ is the model parameter, and τ is the trajectory (e.g., tour) constructed by the model. The goal is to find the best model parameter θ^* to minimize the average total distance (as the training loss $\mathcal{L}$) for τ over a given distribution $p(\mathcal{G})$.

When explicitly considering multiple distributions in model training, most existing works treat each distribution as a task and use meta learning for model training [Manchanda *et al.*, 2023; Zhou *et al.*, 2023]. The objective is to learn a single yet robust model parameter θ^* that can generalize well over various distributions.:

$$\theta^* = \arg\min_{\theta} \frac{1}{T} \sum_{i=1}^{T} \mathbb{E}_{\mathcal{G} \sim p_i(\mathcal{G})} \mathcal{L}\left(\tau \mid \theta, \mathcal{G}\right) \tag{2}$$

where T is the number of tasks, and $p(\mathcal{G})$ represents the distribution over i-th task.

Different from the two approaches, we propose to incorporate prompt learning into the NCO model for tackling cross-distribution vehicle routing problems. The objective can be formulated as:

$$\begin{aligned} &\{P_1^*, \ldots, P_M^*\} \\ &= \arg \min_{\{P_1, \ldots, P_M\}} \frac{1}{T} \sum_{i=1}^{T} \mathbb{E}_{\mathcal{G} \sim p_i(\mathcal{G})} \mathcal{L}\left(\tau \mid P, \theta, \mathcal{G}\right) \end{aligned} \tag{3}$$

where $\{P_1, \ldots, P_M\}$ are M prompts, and P is the selected prompt for each given instance. In this prompt-based model, we can learn the M prompts instead of the entire set of model parameters θ. The objective here is to learn the best prompts that adapt the pre-trained model with a fixed θ for across-distribution performance.

As illustrated in Figure 1, our proposed prompt learning consists of three main components: 1) feature extractor, 2) prompt engineering, and 3) prompted model. We adopt pre-trained attention networks as the feature extractor and the model, which remain fixed during training and testing. The keys are also predetermined based on the features of the randomly generated training instances. The only adjustable components are the prompts. The input instance is fed into both the model and the feature extractor. The feature extractor converts the input instance into a feature vector, allowing us to identify the most appropriate key from the key-prompt pair pool to match the input feature. The key and prompt are coupled together. The associated prompt of the best-matched key is then used to prompt the pre-trained model. A solution is generated by the prompted model, based on the selected prompt. The solution is used to calculate rewards for updating the selected prompt during training.

3.3 Feature Extractor

In this work, we directly use the encoder of the attention model [Kool *et al.*, 2018] as the feature extractor. The encoder consists of L stacked multi-head attention (MHA) blocks. The input of the encoder is the node features $f_i, i = 1, \ldots, n$. In our experiments, the input features for the i-th node are denoted as $f_i = \{x_i, y_i, d_i\}$, where (x_i, y_i) are the coordinates and d_i is the demand. The input features are embedded through a linear projection to generate the initial feature embedding $h_i^{(0)}$. The skip connections [He *et al.*, 2016] and instance normalization (IN) are used in each MHA layer:

$$\begin{aligned} \hat{h}_i^{(l)} &= IN^l\left(h_i^{(l-1)} + MHA_i^l\left(h_1^{(l-1)}, \ldots, h_n^{(l-1)}\right)\right), \\ h_i^{(l)} &= IN^l\left(\hat{h}_i + FF^l\left(\hat{h}_i\right)\right), \end{aligned} \tag{4}$$

where l and $l-1$ represent the current and last embedding layers, respectively. The feedforward (FF) layer contains a hidden sublayer with ReLU activations. The above encoding process generates the final node embeddings $h_i^{(L)}$.

Different from the commonly used feature extraction approach in CV and NLP, which uses the embedding of a specific hidden layer, we concatenate the embeddings from multiple layers. Specifically, we concatenate the output layer of every MHA (before normalization):

$$\begin{aligned} F^l &= \frac{1}{n} \sum_{i=1}^{n} \left(\hat{h}_i^{(l-1)} + MHA^l\left(\hat{h}_i^{(l-1)}\right)\right), \\ F &= cat\{F^1, F^2, \ldots, F^L\}, \end{aligned} \tag{5}$$

where F^l is the hidden embedding before the last norm layer of the l-th MHA and F is the concatenated feature of all L layers. Each hidden embedding F^l is averaged over all n nodes to facilitate generalization across different problem sizes. The final output feature for prompt engineering is adjusted by standard scalarization, given as $F = (F - mean)/stand$, where $mean$ and $stand$ represent the mean and standard deviation of the preprocessing instances, respectively. These preprocessing instances are employed for determining the keys. The mean and standard deviation are calculated element-wise.

3.4 Prompt Engineering

We maintain a key-prompt pair pool, which consists of M key-prompt pairs $\{K_i, P_i\}, i = 1, \ldots, M$, where K_i and P_i are the i-th key and prompt, respectively. Each pair has a fixed key and a learnable prompt. For each input feature F_i, we find the best-matched key $\hat{K} = \min S(F_i, K_j), j = 1, \ldots, M$, where $S()$ is the distance function. The distance function we employ is the Euclidean distance of the input feature and the key. The prompt $\hat{P}$ associated with $\hat{K}$ is then selected to prompt the pre-trained neural solver. In each batch with B instances, B keys are chosen, and the associated B prompts are updated during training.

The keys $K_i, i = 1, \ldots, M$ are predetermined vectors of the same size as the feature. They remain fixed throughout the training process. We randomly sample 128 instances from each of the 341 distributions, resulting in a total of $43,648$ instances for generating the feature data. The 341 distributions are introduced in the Appendix. For each instance i, we utilize the feature extractor introduced in equation (5) to extract the features F_i. We divide the samples into four groups based on problem sizes. For each group, we employ K-means clustering to cluster the samples into N clusters. The cluster centers of the features are then used as the keys. Ultimately, we obtain $M = 4 \cdot N$ vector cluster centers, which are utilized as the keys for prompt learning.

For each key K_i, we randomly initialize a vector as the associated prompt P_i according to a uniform distribution and scale the prompt within the range $[-1, 1]$.

The key-prompt pairs are connected only in terms of utilization, meaning the associated prompts are used based on selected keys. While their values are decoupled, we only update prompts with key fixed during training. The structure is intentionally kept simple, without dynamically adjusting both keys and prompts. Furthermore, the sizes of the keys and pairs are also different. The former is determined by the feature size, while the latter should be sufficiently long to prompt the pre-trained model, which is introduced in the next subsection.

3.5 Prompted Model

We choose the well-known Attention model [Kool *et al.*, 2018; Kwon *et al.*, 2020] as our pre-trained model because it is extensively employed in various routing problems [Bogyrbayeva *et al.*, 2024]. The model consists of a six-layer encoder and a one-layer decoder. During inference, the encoder inferences once, and the solution of the target routing instance is generated iteratively by the decoder. The selected prompts are used for prompting the six-layer encoder, which allows more control over the pre-trained attention model.

Encoder The structure of the pre-trained encoder is the same as that used for the feature extractor. It consists of a six-layer MHA, with the linear projection $h_i^{(0)}$ of instance feature f_i as the input and the final node embedding $h_i^{(L)}$ as the output.

Prompted Encoder The selected prompt P from prompt engineering is firstly split into L subprompts $P^l, l = 1, \ldots, L$. Each subprompt P^l is used for the corresponding embedding layer of the pre-trained encoder. P^l has a length of $D \cdot E$, where D is the number of tokens and E is the length of the token. Then, the l-th subprompt P^l is reshaped into D prompt tokens $p_i^{(l)}, i = 1, \ldots, D$:

$$\begin{aligned} P &= \{P^1, \ldots, P^L\} \\ &= \{p_1^{(1)}, \ldots, p_D^{(1)}, \ldots, p_1^{(L)}, \ldots, p_D^{(L)}\}. \end{aligned} \tag{6}$$

These tokens are concatenated into the corresponding l-th layer of the encoder. Specifically, for the l-th MHA, the D prompt tokens are concatenated with the input hidden layer as follows:

$$\begin{aligned} &\hat{h}_i^{(l)} = IN^l \\ &\left(h_i^{(l-1)} + MHA_i^{(l)} \left(h_1^{(l-1)}, \ldots, h_n^{(l-1)}, \overbrace{p_1^{(l)}, \ldots, p_D^{(l)}}^{\text{D prompt tokens}} \right) \right), \\ &h_i^{(l)} = IN^l \left(\hat{h}_i + FF^l \left(\hat{h}_i \right) \right). \end{aligned} \tag{7}$$

As a result, the length of the input tokens of l-th layer of MHA is always larger than the input tokens of $l - 1$-th layer by D. There will be $n + L \cdot D$ output embedding tokens in the last layer of the encoder. We only use the first output n embedding tokens for the decoder instead of all the $n + L \cdot D$ tokens. The first n embedding tokens represent the n nodes of the instance, which are controlled by the $L \cdot D$ prompt tokens.

Decoder The decoder constructs a solution sequentially. It consists of one MHA layer and one single-head attention (SHA) layer with clipping. The MHA is slightly different from that used in the encoder, where the skip connection, instance normalization, and FF sublayer are now not used [Kool *et al.*, 2018]. The t-th step of inference is as follows:

$$\begin{aligned} \hat{h}_c &= MHA_c \left(h_1^{(L)}, \ldots, h_n^{(L)}, h_t^{(L)}, a_t \right), \\ u_1 \ldots, u_n &= SHA_c \left(h_1^{(L)}, \ldots, h_n^{(L)}, \hat{h}_c \right), \end{aligned} \tag{8}$$

where $h_t^{(L)}$ and a_t represent the embedding of selected node and attribute in the t-th step, respectively. The output embedding of MHA $\hat{h}_c$ is used as the input of the SHA. The SHA outputs the probabilities of choosing the next node using a softmax $p_i = \frac{e^{u_i}}{\sum_j e^{u_j}}$ with the unsatisfied nodes masked. We omit the step indicator t for readability. The detailed structure of the MHA and SHA can be found in Kwon [2020].

Method	Training Cost	50			100			200		
		Dis.	Gap	Time	Dis.	Gap	Time	Dis.	Gap	Time
HGS	/	10.37	0.00%	1.4 h	15.48	0.00%	2.8 h	21.87	0.00%	5.6 h
LKH3	/	10.42	0.49%	1.4 h	15.59	0.69%	2.8 h	22.89	4.69%	16.7 h
POMO	/	10.98	5.92%	1.5 s	15.82	2.18%	2.7 s	23.27	6.41%	17 s
Meta POMO	>3 d	10.77	3.89%	1.5 s	16.15	4.28%	2.9 s	23.14	5.83%	18 s
Omni	3 d	10.99	5.98%	1.5 s	16.04	3.58%	2.9 s	22.80	4.29%	18 s
Prompt	1 d	10.70	3.20%	1.5 s	15.88	2.57%	2.9 s	22.65	3.58%	18 s
Prompt top-8	1 d	**10.63**	**2.51%**	12 s	**15.78**	**1.94%**	23 s	**22.58**	**3.25%**	2.4 m
POMO aug	/	10.72	3.40%	5 s	15.69	1.36%	16 s	23.00	5.19%	86 s
Meta POMO aug	>3 d	10.60	2.22%	5 s	15.96	3.08%	16 s	22.90	4.75%	88 s
Omni aug	3 d	10.75	3.69%	5 s	15.86	2.43%	16 s	22.63	3.51%	88 s
Prompt aug	1 d	10.54	1.67%	5 s	15.74	1.65%	16 s	22.46	2.74%	89 s
Prompt top-8 aug	1 d	**10.51**	**1.31%**	40 s	**15.68**	**1.26%**	2.1 m	**22.43**	**2.56%**	12 m

Table 1: Comparison of different methods on three training distributions.

3.6 Training with Reinforcement Learning

We use the REINFORCE algorithm with a shared baseline following Kwon [2020] to update the selected prompts in each batch. We use greedy inference, i.e., a deterministic trajectory τ is constructed iteratively based on the policy. In each construction step t, the next node v_t is selected as the one with the maximum softmax probability $t = \arg\max_i(p_i)$ predicted by the decoder. n trajectories are constructed from n different starting points.

The rewards $R(\tau_1), \dots, R(\tau_n)$ are the negative total distances of trajectories $\tau_1, \dots, \tau_n$. We use the following gradient ascent to update prompts P in each batch with size B:

$$\nabla_P J(\theta, P) \approx \frac{1}{nB} \sum_{i=1}^{B} \sum_{j=1}^{n} \left(R\left(\tau_j^i \mid s\right) - b(s) \right) \nabla_P \log p_{\theta,P}\left(\tau_j^i \mid s\right), \tag{9}$$

where P and θ are trained prompts and fixed parameters for the model. s represents the instances. $p_{\theta,P}(\tau_j^i)$ is the aggregation of the probability of selection in each step of the decoder based on both the fixed θ and the prompts P. $b(s)$ is the shared baseline [Kwon *et al.*, 2020].

4 Experiments

4.1 Experimental Setting

Model Settings We use the Attention model as our backbone pre-trained model. It is only trained on uniformly distributed CVRP instances of size 100. All the settings for the pre-trained model are the same as that used in the paper of Kwon [2020]. The number of encoder MHA layers is six and the decoder consists of one MHA and one SHA.

The settings of the key-prompt pair pool are as follows: The prompt size is set to be $M = 16$, and the number of prompted tokens for each layer is $D = 5$. As there are $L = 6$ MHA layers in the encoder and the embedding size for each token is $E = 128$, the lengths of the key and prompt vectors are $6 \cdot 128 = 768$ and $5 \cdot 6 \cdot 128 = 3840$, respectively.

Instance Generation We trained the model on a set of routing tasks with diverse sizes and geometrical distributions. The detailed instance generation is introduced in the Appendix, which is the same as that used by Zhou [2023]. The problem size ranges from 50 to 200 with both uniform distribution and various clustered Gaussian distributions. There are in total 341 distributions.

Training Setup The prompts are trained with a batch size of $B = 64$. The 341 distributions are sequentially used during the training. In each batch, we randomly sample B instanced from one distribution. As a result, every distribution will be sampled in 341 epochs. The maximum number of epochs is $10,000$ and there are $1,000$ instances for each epoch. The learning rate decays from $1e-3$ to $1e-5$. It takes only about 2.5 days on a single RTX 2080Ti with 11 GB GPU memory.

Baselines We compared our proposed prompt learning to three different types of methods. 1) Baseline heuristic VRP solvers: hybrid genetic search (HGS) [Vidal *et al.*, 2013], LKH3 [Helsgaun, 2017]; 2) NCO methods: basic POMO [Kwon *et al.*, 2020] trained on single-distribution, and POMO [Zhou *et al.*, 2023] trained on diverse distribution; 3) meta learning NCO methods: Meta-POMO [Manchanda *et al.*, 2023], and the newest revision of meta-learning method for omni-generalization on vehicle routing problems (Omni) [Zhou *et al.*, 2023]. The results for HGS and LKH are obtained by executing the open-source code on the instances. For the basic POMO, we train the model by ourselves on CVRP of size 100 with uniform distribution. This model is also utilized as the pre-trained model in our proposed prompt learning approach. For the meta-learning methods, we utilized the pre-trained model from Zhou [2023] as it was trained on the same distributions as ours.

4.2 Results on Training Tasks

The results on training distributions are compared in Table 1. For simplicity, we list the averaged results over 1,000 instances on three problem sizes with uniform distribution. More results are included in the Appendix. We use HGS as the baseline and compare the training cost, average distances (Dis.), average gap to the baseline (Gap), and the inference time cost (Time). We executed HGS and LKH with time limits of 5s and 10s for problem sizes of 50 and 100, respectively. For instances with a problem size of 200, the time

	50 CL	50 EA	50 EO	50 IM	50 GR	50 MX	100 CL	100 EA	100 EO	100 IM	100 GR	100 MX	Avg.
HGS	0	0	0	0	0	0	0	0	0	0	0	0	0
LKH3	1.66%	1.53%	1.69%	1.81%	1.70%	1.62%	1.79%	1.61%	2.21%	3.61%	3.58%	3.80%	2.22%
Meta POMO	4.14%	3.56%	3.87%	3.95%	3.93%	3.58%	4.34%	3.73%	4.29%	4.32%	4.17%	3.80%	3.97%
Omni	4.61%	4.71%	5.20%	5.76%	5.63%	5.10%	3.32%	3.07%	3.81%	3.80%	3.67%	3.63%	4.36%
Prompt	**3.93%**	**2.98%**	**3.12%**	**3.26%**	**3.23%**	**3.25%**	**3.75%**	**2.64%**	**2.88%**	**2.67%**	**2.52%**	**3.01%**	**3.10%**
Meta POMO aug	2.33%	2.05%	2.10%	2.24%	2.18%	2.00%	2.97%	2.48%	3.05%	3.07%	2.95%	2.60%	2.50%
Omni aug	2.74%	2.80%	3.09%	3.51%	3.50%	2.96%	2.26%	1.98%	2.61%	2.66%	2.54%	2.46%	2.76%
Prompt aug	**1.97%**	**1.48%**	**1.51%**	**1.63%**	**1.66%**	**1.63%**	**2.36%**	**1.50%**	**1.82%**	**1.68%**	**1.56%**	**1.89%**	**1.72%**

Table 2: Zero-shot generalization performance on 12 new distributions.

costs for each instance with HGS and LKH were 20s and 60s, respectively. It should be noted that LKH3 is not fully converged in some instances. We adopted the same data augmentation method (aug) as in Kwon [2020]. Additionally, for our prompt learning, we present the inference results using the top-k matched prompts. Further discussion on the top-k prompts is provided in the following section.

The proposed prompt learning method has a considerable reduction in training costs when compared to meta-learning methods. According to Zhou [2023], the second-order derivative method needs about 6 days and 71GB GPU memory and the training cost can be reduced to 3 days on 25GB GPU with a smarter strategy. For a fair comparison, we adjust the training cost of our prompt learning approach to match the experimental settings of the meta-learning methods. Specifically, prompt learning requires roughly 1 day on a 25GB GPU, considering the allowance for a larger batch size.

Our prompt learning model outperforms existing meta-learning methods and the basic POMO model on all three problem sizes in terms of average distances. The basic POMO model with single-distribution learning overfits the training distribution. Because the basic POMO is trained on uniform distribution instances with a size of 100, it has good performance on the 100 instances set but deteriorates on the other two sizes. The two meta-learning methods' performance is more robust across different sizes compared with the basic POMO. Our prompt learning further reduces the gap. The prompt learning with the top 8 prompts ranks first in all sizes.

4.3 Zero-shot Generalization

We demonstrate the zero-shot generalization performance of prompt learning on new distributions that were not used during training. We adopt the distribution proposed by Bi [2022], which consists of a total of 12 different distributions, including cluster (CL), expansion (EA), explosion (EO), implosion (IM), grid (GR), and mixed (MX). Each distribution encompasses two different problem sizes, and we conduct tests on 1,000 instances for each distribution. We use HGS as the baseline. The total execution times on each distribution for both HGS and LKH on sizes 50 and 100 are 1.4h and 2.8h, respectively.

Table 2 lists the zero-shot generalization performance on the 12 new distributions. The best results are in bold. Our prompt learning achieves the best average results. The average gap to the baseline is about 3%. With data augmentation, the gap can be further reduced to less than 2%.

4.4 Top-k Prompts

Instead of relying on a single best-matched prompt, we can employ multiple prompts simultaneously during the inference stage to enhance performance. To achieve this, we propose a top-k strategy, in which the top-k prompts (determined by the Euclidean distance between the key and feature vectors) are chosen. These k prompts are then used concurrently during inference, and the best solution is selected as the final solution for each instance. This approach allows us to fully leverage our learned prompts without incurring any additional training costs.

Figure 2 shows the results obtained on instances of three different problem sizes with uniform distribution. The x-axis represents the number of prompts (k) employed in the top-k strategy, while the y-axis represents the difference in performance compared to the baseline HGS. Generally, the performance improves with an increase in the number of prompts. Moreover, the reduction in the gap is not linearly proportional to the number of prompts used, which suggests that the best-matched prompt is more significant than others.

4.5 Prompt Token Size

The number of prompt tokens D in each encoder layer influences the performance of our prompt learning network. A larger number of tokens results in a longer prompt vector, providing the ability to prompt the attention-based encoder more effectively. In order to investigate the impact of the token number on our models, we conducted two additional prompt learning experiments. Specifically, we set the token numbers in the two models as 1 and 10, respectively. Consequently, the lengths of the prompt vectors in these models are $1 \cdot 6 \cdot 128 = 768$ and $10 \cdot 6 \cdot 128 = 7680$. All other training models and settings remain unchanged.

The outcomes of the experiments involving different numbers of prompt tokens are presented in Table 3. The table includes results from four training distributions, distinguished by their numbers of prompt tokens. U and GM represent uniform distribution and Gaussian distribution with 3 clusters and scaled by 50 (details of instance generalization please refer to the Appendix). Minor differences in results are ob-

	50 U	100 U	200 U	200 GM_50_3
HGS	0	0	0	0
Token 1	3.84%	2.46%	4.18%	4.45%
Token 1 aug	2.00%	1.54%	3.27%	2.93%
Token 5	3.20%	2.57%	3.58%	3.30%
Token 5 aug	1.67%	1.65%	2.74%	2.24%
Token 10	2.99%	2.59%	3.43%	2.97%
Token 10 aug	**1.47%**	**1.67%**	**2.60%**	**2.02%**

Table 3: Results with different prompt token sizes.

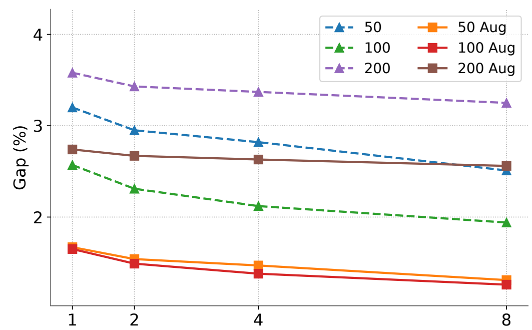

Figure 2: Results with different numbers of top-k prompts.

	POMO	Meta POMO	Omni	Prompt	Prompt top-8
A	7.3%	2.3%	4.4%	2.1%	**1.8%**
B	12.6%	1.9%	2.4%	1.7%	**1.5%**
P	35.6%	12.9%	10.8%	3.8%	**2.7%**
X	5.4%	4.9%	4.9%	4.7%	**3.5%**
XML	4.4%	5.4%	5.8%	6.1%	**3.4%**
Average	13.2%	5.4%	5.6%	3.5%	**2.5%**

Table 4: Results on CVRPLib Instances.

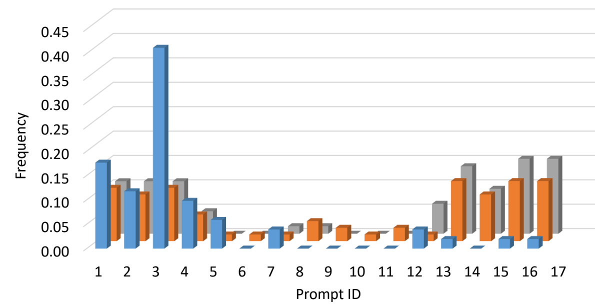

Figure 3: Selection frequencies of prompts on three different test sets. Blue: Set P, Orange: Set X, Grey: Set XML.

served for instances with a size of 100, while more significant variations are noticed across other distributions. This discrepancy arises because the pre-trained basic POMO model utilized in prompt learning is trained on routing instances with a size of 100. Hence, the model already exhibits satisfactory performance on in-distribution instances. However, when adapting the pre-trained model to out-of-distribution instances, the number of tokens assumes importance. For example, in the case of 200 GM instances, there is approximately a 1% performance increase (reduction in gap) from 1 token (2.93%) to 10 tokens (2.02%). Overall, prompt learning with a larger token size allows better generalization performance.

4.6 Real-world Instances

More experiments on real-world instances are conducted on five test suites: set A, B, P, X [Uchoa *et al.*, 2017], and XML [Queiroga *et al.*, 2021] from CVRPLIB [1]. There are 115 instances in total with various geometrical distributions, demands, and problem sizes, which can provide a comprehensive evaluation of our proposed method. Table 4 summarizes the average gap to the best-known results from CVRPLIB. The best results are in bold. The detailed results can be found in the Appendix.

In Figure 3, we visualize the selected frequencies (normalized) of 16 prompts on test set P, X, and XML. Set X and Set XML have similar frequency distributions, which are different from Set P. For example, prompts 11-15 are frequently used for Set X and XML while rarely used in Set P. The results show that the instances from Set X and Set XML are of similar distributions. As has been mentioned in the paper of Queiroga [2021], the generator of XML is the same one used for X instances. It answers why the basic POMP performs well on these two sets while much worse on Set P. It also demonstrates that our prompt learning can recognize the features of new instances and select the best-matched prompt for better performance.

5 Conclusion

This paper investigates the first prompt learning based neural combinatorial optimization (NCO) method to solve vehicle routing problems over diverse distributions. We propose a prompt-based attention network with a learnable key-prompt pair pool to facilitate the fast zero-shot adaptation of the pre-trained NCO model for cross-distribution generalization. To evaluate the effectiveness of our proposed prompt learning method, we conduct extensive experiments on test instances with various distributions. The results clearly demonstrate the superiority of our approach over classical single-distribution learning methods and existing meta learning techniques. Our prompt-based model achieves improvements in both in-distribution prediction and zero-shot generalization to a diverse set of new tasks while requiring lower training costs.

Acknowledgments

The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (GRF Project No. CityU 11215723) and the Shenzhen Technology Plan, China (Grant No. JCYJ20220530113013031).

[1]http://vrp.atd-lab.inf.puc-rio.br/

References

[Bai *et al.*, 2023] Ruibin Bai, Xinan Chen, Zhi-Long Chen, Tianxiang Cui, Shuhui Gong, Wentao He, Xiaoping Jiang, Huan Jin, Jiahuan Jin, Graham Kendall, et al. Analytics and machine learning in vehicle routing research. *International Journal of Production Research*, 61(1):4–30, 2023.

[Bello *et al.*, 2016] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[Bengio *et al.*, 2021] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[Bi *et al.*, 2022] Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. Learning generalizable models for vehicle routing problems via knowledge distillation. *arXiv preprint arXiv:2210.07686*, 2022.

[Bogyrbayeva *et al.*, 2024] Aigerim Bogyrbayeva, Meraryslan Meraliyev, Taukekhan Mustakhov, and Bissenbay Dauletbayev. Machine learning to solve vehicle routing problems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Chen and Tian, 2019] Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[Drakulic *et al.*, 2023] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for generalizable neural combinatorial optimization. *arXiv preprint arXiv:2301.03313*, 2023.

[Fu *et al.*, 2021] Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7474–7482, 2021.

[Ge *et al.*, 2023] Chunjiang Ge, Rui Huang, Mixue Xie, Zihang Lai, Shiji Song, Shuang Li, and Gao Huang. Domain adaptation via prompt learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2023.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Helsgaun, 2017] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12, 2017.

[Hottung and Tierney, 2019] André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. *arXiv preprint arXiv:1911.09539*, 2019.

[Jia *et al.*, 2022] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022.

[Jiang *et al.*, 2022] Yuan Jiang, Yaoxin Wu, Zhiguang Cao, and Jie Zhang. Learning to solve routing problems via distributionally robust optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9786–9794, 2022.

[Jiang *et al.*, 2023] Yuan Jiang, Zhiguang Cao, Yaoxin Wu, and Jie Zhang. Multi-view graph contrastive learning for solving vehicle routing problems. In *Uncertainty in Artificial Intelligence*, pages 984–994. PMLR, 2023.

[Joshi *et al.*, 2022] Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27(1-2):70–98, 2022.

[Kool *et al.*, 2018] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[Kool *et al.*, 2022] Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 19th International Conference, CPAIOR 2022, Los Angeles, CA, USA, June 20-23, 2022, Proceedings*, pages 190–213. Springer, 2022.

[Kwon *et al.*, 2020] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[Li *et al.*, 2022] Bingjie Li, Guohua Wu, Yongming He, Mingfeng Fan, and Witold Pedrycz. An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem. *IEEE/CAA Journal of Automatica Sinica*, 9(7):1115–1138, 2022.

[Liu *et al.*, 2023] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[Manchanda *et al.*, 2023] Sahil Manchanda, Sofia Michel, Darko Drakulic, and Jean-Marc Andreoli. On the generalization of neural combinatorial optimization heuristics. In *Machine Learning and Knowledge Discovery in*

Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part V, pages 426–442. Springer, 2023.

[Nazari *et al.*, 2018] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

[Ouyang *et al.*, 2022] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[Pan *et al.*, 2023] Xuanhao Pan, Yan Jin, Yuandong Ding, Mingxiao Feng, Li Zhao, Lei Song, and Jiang Bian. H-tsp: Hierarchically solving the large-scale traveling salesman problem. In *AAAI 2023*, February 2023.

[Queiroga *et al.*, 2021] Eduardo Queiroga, Ruslan Sadykov, Eduardo Uchoa, and Thibaut Vidal. 10,000 optimal cvrp solutions for testing machine learning based heuristics. In *AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR)*, 2021.

[Raza *et al.*, 2022] Syed Mohib Raza, Mohammad Sajid, and Jagendra Singh. Vehicle routing problem using reinforcement learning: Recent advancements. In *Advanced Machine Intelligence and Signal Processing*, pages 269–280. Springer, 2022.

[Toth and Vigo, 2014] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[Uchoa *et al.*, 2017] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

[Vidal *et al.*, 2013] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research*, 40(1):475–489, 2013.

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

[Wang *et al.*, 2023] Ning Wang, Jiahao Xie, Jihao Wu, Mingbo Jia, and Linlin Li. Controllable image captioning via prompting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2617–2625, 2023.

[Xu *et al.*, 2022] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pages 24631–24645. PMLR, 2022.

[Zhang *et al.*, 2023] Renrui Zhang, Xiangfei Hu, Bohao Li, Siyuan Huang, Hanqiu Deng, Yu Qiao, Peng Gao, and Hongsheng Li. Prompt, generate, then cache: Cascade of foundation models makes strong few-shot learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15211–15222, 2023.

[Zhou *et al.*, 2022] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[Zhou *et al.*, 2023] Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Towards omni-generalizable neural methods for vehicle routing problems. In *the 40th International Conference on Machine Learning (ICML 2023)*, 2023.