Learning to Solve Routing Problems via Distributionally Robust Optimization

Yuan Jiang[1*], Yaoxin Wu[1*], Zhiguang Cao[2†], Jie Zhang[3]
[1]SCALE@NTU Corp Lab, Nanyang Technological University, Singapore
[2]Singapore Institute of Manufacturing Technology, A*STAR, Singapore
[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore
jiang.yuan@ntu.edu.sg, wuyaoxin@ntu.edu.sg, zhiguangcao@outlook.com, zhangj@ntu.edu.sg

Abstract

Recent deep models for solving routing problems always assume a single distribution of nodes for training, which severely impairs their cross-distribution generalization ability. In this paper, we exploit group distributionally robust optimization (group DRO) to tackle this issue, where we jointly optimize the weights for different groups of distributions and the parameters for the deep model in an interleaved manner during training. We also design a module based on convolutional neural network, which allows the deep model to learn more informative latent pattern among the nodes. We evaluate the proposed approach on two types of well-known deep models including GCN and POMO. The experimental results on the randomly synthesized instances and the ones from two benchmark dataset (i.e., TSPLib and CVRPLib) demonstrate that our approach could significantly improve the cross-distribution generalization performance over the original models.

Introduction

Combinatorial optimization problems (COPs) with NP-hardness are always featured by discrete search space and intractable computation to seek the optimal solution. As a fundamental COP in logistics, the vehicle routing problem (VRP) (Dantzig and Ramser 1959) concerns the cost-optimal delivery of items from the depot to a set of geographically scattered customers through vehicle(s). It has been extensively investigated for decades and found widespread applications in reality, such as waste collection (Han and Ponce Cueto 2015), dial-a-ride (Malheiros et al. 2021) and courier delivery (Steever, Karwan, and Murray 2019).

The studies on VRP with deep (reinforcement) learning is emerging in recent years. Different from the conventional methods, this line of works aims at automatically searching heuristic policies by using neural networks to learn the underlying patterns in instances, which could be used to discover better policies than hand-crafted ones (Bengio, Lodi, and Prouvost 2021). Towards reducing the gaps to the highly optimized conventional heuristic solvers including Concorde (Applegate et al. 2006) and LKH (Helsgaun 2000), a large number of efforts have been performed to invent various deep models to solve the VRP variants, i.e., traveling salesman problem (TSP) and capacitated vehicle routing problem (CVRP) (Khalil et al. 2017; Kool, van Hoof, and Welling 2019; Chen and Tian 2019; Hottung and Tierney 2020; Ma et al. 2021; Wu et al. 2021; Kwon et al. 2020; Li et al. 2021; Xin et al. 2021b).

Although much success has been achieved, existing deep models always assume a pure spatial distribution of nodes (customers) for training, i.e., uniform distribution, which severely limits their applications given the impaired cross-distribution generalization ability. While it is natural to stipulate that a majority of typical instances follow an identical distribution, a minority of atypical (yet important) instances which follow different one(s) may always exist in reality (Hashimoto et al. 2018). In this sense, the mono-training paradigm in existing deep models may cause inferior performance or even failures for those atypical instances. E.g., a trained routing policy may offer unreasonable routes to serve a group of (important) customers whose location pattern differs from the ones used in training. Intuitively, an alternative is to force the model to homogeneously treat all the instances of different distributions for training. However, it does not necessarily enhance the cross-distribution generalization performance in our view, as it may suffer high losses on the group of the atypical instances.

To address this issue, we propose an approach by exploiting *group distributionally robust optimization* (group DRO) to jointly train the deep models on instances of different groups (more than one group), where each group follows a distribution. In particular, we aim at minimizing the loss for the *worst-case* group during training, where we optimize the weights for different groups of instances and the parameters for the deep model in an interleaved manner. More importantly, we do not need to label the distribution for each group during inference. In addition, we also leverage convolution neural network (CNN) to learn initial representations of VRP instances so that the distribution-aware features in spatial patterns could be favorably captured to boost the performance further. The experimental results show that, our approach not only achieves superior performances on the overall instances and the worst-case performance on the

[*]Yuan Jiang and Yaoxin Wu contributed equally.
[†]Zhiguang Cao is the corresponding author.

atypical instances, but also exhibits desirable applicability to different deep models. Note that we do not claim to surpass the state-of-the-art deep methods for solving VRPs in all aspects, but to inject them with robustness for better cross-distribution generalization instead. Our contributions in this paper are summarized as follows.

- We exploit group DRO to add the dimension of robustness to deep models, which enhances their cross-distribution generalization for solving VRPs. The proposed approach could be freely applied to various deep models, in the fashions of either supervised or reinforcement learning, e.g., GCN (Joshi, Laurent, and Bresson 2019) and POMO (Kwon et al. 2020), respectively.
- We leverage CNN to initially identify the spatial pattern of the nodes in VRP instances, which allows the deep models to learn more informative distribution-aware representation and thus generate solutions of higher quality.
- We apply our approach to solve randomly generated instances of different distributions. The results show that it not only improves the overall performance and the worst-case performance over the original deep models, but also achieves superior performance for solving the instances from the benchmark dataset, i.e., TSPLib and CVRPLib.

Related Works

In this section, we provide a review of the deep models for solving VRPs, and the DRO in the machine learning community, respectively.

Deep Models for VRPs

The recent learning-based methods have shown great merits to automatically discover heuristics for solving VRPs, which effectively circumvent massive human expertise and trial-and-error required in the classic hand-engineering methods. Most of the deep models concentrate on solving TSP and CVRP, i.e., two representative VRP variants. The very early endeavor was the Pointer Network which used a recurrent neural network (RNN) to learn node selection in a supervised manner (Vinyals, Fortunato, and Jaitly 2015). Similarly, Joshi, Laurent, and Bresson (2019) predicted edges which will appear in the optimal solutions with supervised learning, where a graph convolutional network (GCN) was developed for deep embedding of both nodes and edges. The other works diverged mainly by employing different Transformer-based architectures and training with reinforcement learning (Bello and Pham 2017; Kool, van Hoof, and Welling 2019; Xin et al. 2021a). Specially, Khalil et al. (2017) tackled TSP using a graph embedding network. Instead of learning heuristics to select nodes, another line of works sought policies to improve solutions with local search frameworks (Chen and Tian 2019; Lu, Zhang, and Yang 2019; Hottung and Tierney 2020; Wu et al. 2021). Among the deep models, Kwon et al. (2020) presented a method called POMO to train multiple rollouts on augmented instances and delivered the state-of-the-art performance. On the other hand, despite the near-optimal results, most of the above deep models are validated with the synthesized instances of the uniform distribution. The cross-distribution generalization is not explicitly considered in their training process, which may degenerate on non-uniform instances, e.g., the ones from the well-known TSPLib and CVRPLib.

Distributionally Robust Optimization

It is known that the standard maximum likelihood estimation may degrade the inference performance of deep models, if the training samples are out of the distribution for test (Oren et al. 2019; Kuhn et al. 2019). This issue about the out-of-distribution generalization is still stubborn for the general learning models, either in supervised or reinforcement learning (Sun et al. 2020; Cobbe et al. 2019).

One of the solutions to ameliorate the generalization capability is using distributionally robust optimization (DRO), which seeks to minimize losses over all sub-populations of the training distribution (Ben-Tal et al. 2013; Duchi, Hashimoto, and Namkoong 2019). It was originally designed for classic underparameterized models to reduce the training loss, by regularizing the models and defending them against adversarial examples (Namkoong and Duchi 2017; Sinha, Namkoong, and Duchi 2018). Different from them, group DRO instead defines the uncertainty set as mixtures (or combinations) of groups over training data to avoid optimizing implausible worst-case groups (Hu et al. 2018; Oren et al. 2019). In this line, Sagawa et al. (2019) applied group DRO in the overparameterized regime with vanishing training loss and poor worst-case generalization, and suggested that desirable accuracy for the worst-case group could be attained even if the groups are imperfectly designated. In this paper, we exploit group DRO to enhance the cross-distribution generalization of deep models for solving VRPs, where we explicitly define the uncertainty via (instance) groups of different distributions and minimize the worst expected loss over them.

Preliminaries

In this section, we present the graph representation of TSP and CVRP, followed by the basic rationale of DRO.

VRPs and Data

We consider TSP and CVRP in 2D Euclidean space. A TSP instance is represented as a fully connected graph G with n nodes $\{v_1, v_2, \ldots, v_n\}$, where the edge weights correspond to distances between nodes. We target at the common objective to find a permutation of the nodes π, i.e., a tour, which traverses each node once and returns to the starting one with the shortest distance. On top of the TSP graph, CVRP requires one more node v_0 as the depot, and prescribes the demands $\{\delta_1, \delta_2, \ldots, \delta_n\}$ for each node. The objective is to decide routes with the shortest distance for the vehicle(s) satisfying that, 1) each route starts and ends at the depot; 2) each node is traversed by one route; 3) the sum of demands in a route is less than the pre-defined capacity D of vehicle.

Learning-based methods often assume a uniform distribution of nodes to generate the instances, and simply pursue smallest average (optimality) gap of the objective values over them. It inevitably sacrifices the performance on atypical instances that do not follow a majority distribution (i.e.,

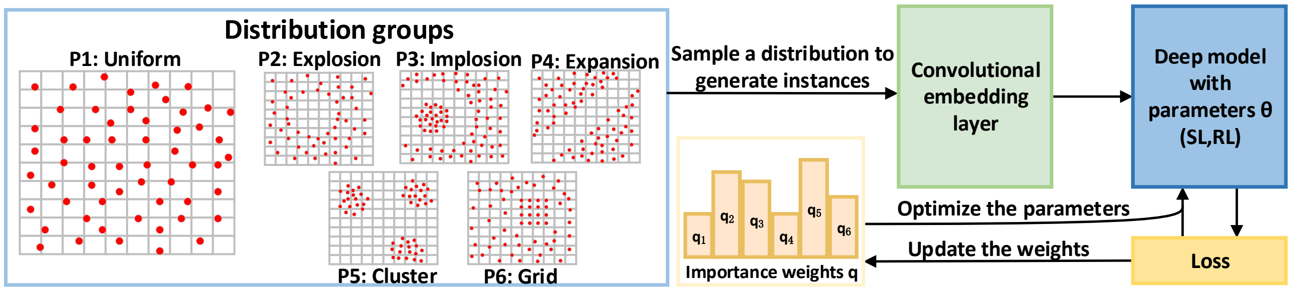

Figure 1: An example illustration of our approach with VRP instances sampled from six distribution groups, which alternately optimizes the parameters θ of a deep model and updates the importance weights q of the distribution groups in the training set.

when more than one distribution exist), which may also deteriorate the overall performance.

Distributionally Robust Optimization

Given a parameter family Θ, loss function ℓ and training samples x drawn from a distribution P, most of existing deep models directly optimize the empirical risk minimization (ERM) as below,

$$\hat{\theta}_{\text{ERM}} := \arg\min_{\theta\in\Theta} \mathbb{E}_{x\sim\hat{P}}[\ell(\theta;x)], \tag{1}$$

where $\hat{P}$ is the empirical distribution over the training samples. When the test distribution is same as the one for training, it guarantees that a model trained via ERM performs well for test given sufficient training samples. However, ERM may impair the performance on atypical samples since the model is tuned to emphasize more on the majority samples of the distribution for training.

Different from ERM, DRO aims at optimizing parameters θ for a model to achieve more accurate predictions over the test set following diverse or even unknown distributions. It hinges on a more conservative objective that encourages the model to assign higher weights to optimize for atypical samples. Formally, DRO minimizes the *worst* expected loss over an *uncertainty set* of distributions $\mathcal{Q}$ as below,

$$\min_{\theta\in\Theta} \left\{ \mathcal{R}(\theta) := \sup_{p\in\mathcal{Q}} \mathbb{E}_{x\sim p}[\ell(\theta;x)] \right\}, \tag{2}$$

where $\mathcal{Q}$ includes the potential test distributions that we hope the model to perform well on, and p is a possible training distribution sampled form the uncertainty set. By minimizing the *worst-case* loss for all distributions in the uncertainty set $\mathcal{Q}$, we can also expect to achieve desirable overall performance. Note that the objective in Eq. (2) does not necessarily rely on the unknown test distribution p'. In the meantime, it is also the upper bound of the test risk (Oren et al. 2019), i.e., $\mathbb{E}_{p'}[\ell(x;\theta)] \leq \sup_{p\in\mathcal{Q}} \mathbb{E}_p[\ell(\theta;x)]$.

Methodology

In this section, we first exploit the group DRO to train deep models for solving VRPs, which allows them to favorably handle instances of different distributions. Then we fill the gaps for deep models that hinge on reinforcement learning (RL). Finally, we present a convolutional module which could effectively initialize the distribution-aware representations for VRP instances. The overview of our approach is illustrated in Figure 1.

Group DRO for VRPs

We stipulate that the VRP instances in the training set follow a mixture of distributions rather than a single one, e.g., the majority of typical instances are generated *uniformly* along with the minority of atypical ones sampled from another distribution. While different from those in existing deep models, this setting offers two merits that, 1) it explicitly considers heterogeneous instances, where atypical (yet important) ones may always exist in reality even on a daily basis; 2) the cross-distribution generalization ability could be potentially enhanced. To train deep models with the above setting, DRO is a conceptually appealing option to balance the training on instances of different distributions, since it minimizes the worst expected loss over different distributions in the uncertainty set. It will also potentially empower deep models to tackle the discrepancy between the training and test distributions. However, the effectiveness of DRO often hinges on the choice of the uncertainty set $\mathcal{Q}$, where common practice is to define it as a divergence ball over the entire training distribution. Unfortunately, it may lead to an overly conservative set of potential test distributions that overemphasizes on the minority instances.

To circumvent this issue, we exploit the *group* DRO and explicitly leverage prior knowledge of distributions to group the instances in the training set. In this way, the inferior models that are overwhelmed by unreasonable worst-case distributions due to the divergence ball in DRO, will be avoided (Duchi, Hashimoto, and Namkoong 2019). Particularly, we allocate training instances of each distribution into a respective group, termed *distribution group*, and the uncertainty set $\mathcal{Q}$ is defined as the combination of these distribution groups as below,

$$\mathcal{Q} := \left\{ \sum_{g=1}^{m} q_g P_g : q \in \Delta_m \right\}, \tag{3}$$

Algorithm 1: Group DRO for Solving VRPs

Input: Training set S, hyperparameter for $M(\theta; x)$
Output: Model parameters θ

```
1:  Initialize model parameters $\theta^{(0)}$
2:  Initialize group weights $q^{(0)}$
3:  for t = $1, 2, \ldots, T$ do
4:      Sample a group index $g$ from $(1, \ldots, m)$
5:      for $t' = 1, 2, \ldots, T'$ do
6:          Sample a batch of instances $s$ from group $g$
7:          Sample trajectories via $M(\theta^{(t')}; x)$ solving $s$
8:          Calculate the reinforce loss $\ell_r$
9:          $\theta^{(t')} \leftarrow \theta^{(t'-1)} - \eta_\theta q_g^{(t)} \nabla \ell_r \left(\theta^{(t'-1)}; (x, g)\right)$
10:     end for
11:     $q' \leftarrow q^{(t-1)}; q'_g \leftarrow q'_g \exp\left(\eta_q \ell_r \left(\theta^{(t')}; (x, g)\right)\right)$
12:     Renormailize weights by $q^{(t)} \leftarrow q' / \sum_{g'} q'_{g'}$
13: end for
```

where the training set is a mixture of m distribution groups P_g, indexed by $\mathcal{G} = \{1, 2, \ldots, m\}$; Δ_m denotes the $(m-1)$-dimensional probability simplex; q_g denotes the *importance weight* of the g-th distribution group.

In light of the above setting, each training instance is associated with a 2-tuple (x, g), where x denotes the input to deep models for an instance and g is its group index. While the group index of each instance in training will enhance the deep models with more desirable cross-distribution generalization ability, this group index is not required during inference. Formally, we update the parametrized model by minimizing the worst empirical expected loss as below,

$$\hat{\theta}_{\text{DRO}} := \underset{\theta \in \Theta}{\arg\min} \left\{ \hat{\mathcal{R}}(\theta) := \max_{g \in \mathcal{G}} \mathbb{E}_{x \sim \hat{P}_g} [\ell(\theta; x)] \right\}, \quad (4)$$

where $\hat{\mathcal{R}}(\theta)$ refers to the maximum empirical expected loss among all distribution groups, and each group $\hat{P}_g$ is an empirical distribution over the corresponding training instances. The above equation could be further reformulated as below,

$$\min_{\theta \in \Theta} \sup_{q \in \Delta_m} \sum_{g=1}^{m} q_g \mathbb{E}_{(x) \sim P_g} [\ell(\theta; x)], \quad (5)$$

where q is a m-dimensional trainable vector with the same meaning as the one in Eq. (3). According to Eq. (5), we propose to optimize the parameters of the deep model and importance weights of the distribution groups in an interleaved manner. Specifically, we train a deep model to minimize the loss $\hat{\mathcal{R}}(\theta)$ over distribution groups, while importance weight q_g is updated and used to emphasize the expected losses for each distribution group g.

Adapt DRO with RL The DRO with the fashion of supervised learning has been widely studied in the machine learning community (Namkoong and Duchi 2017; Oren et al. 2019; Sagawa et al. 2019), which could be naturally adapted with the deep models trained in a supervised way for solving VRPs, such as GCN (Joshi, Laurent, and Bresson 2019). Compared with supervised learning, reinforcement learning (RL) is much preferred given its independence of optimal solution as the ground-truth label. For example, POMO (Kwon et al. 2020) trained with reinforcement learning, has achieved state-of-the-art performance among all the deep models. As such, we elaborate on how to adapt the DRO with RL algorithms.

Without loss of generality, we define a deep model M to sample solutions to VRPs, i.e., $P_\theta(\pi|x) = M(\theta; x)$, where θ denotes trainable parameters for M; x and π denote the input and the solution, respectively. We extend REINFORCE (Williams 1992) to encourage M to generalize well across different distributions. The reinforce loss ℓ_r for VRPs is the expected length of the routes in the solution, i.e., $l(\pi)$. Accordingly, we train the deep model parameters θ using the gradient of the worst expected reinforce loss as below,

$$\nabla \ell = \sup_{q \in \Delta_m} \sum_{g=1}^{m} q_g \mathbb{E}_{\substack{M(\theta;x) \\ x \sim P_g}} [(l(\pi) - b(x)) \nabla log M(\theta; x)], \quad (6)$$

where $b(x)$ denotes the baseline to reduce the variance of gradients. We maintain a importance weight q_g for the distribution group P_g. The hybrid training with DRO and RL is summarized in Algorithm 1. Particularly, following the DRO for supervised learning (Sagawa et al. 2019), we alternately optimize model parameters θ using stochastic gradient descent (SGD) with the fixed importance weights q (line 9), and update q using exponentiated gradient ascent (line 11). Additionally, we fix q for every T' iterations to stabilize the training (lines 5-10). Note that this algorithm can be applied to a wide range of deep RL models for VRPs.

Distribution-aware Embedding via CNN

On the one hand, the nodes in a VRP solution are always connected with the ones in their close proximities, and the local spatial information is critical to reveal the node distribution, which is helpful to enhance the cross-distribution generalization performance for the deep models. On the other hand, while the convolutional neural networks (CNNs) have demonstrated strong capability for learning spatial features in computer vision, directly mapping the graph representation of VRP instances to 2-dimensional images and then applying CNN did not bring obvious benefit (Miki and Ebara 2019; Ling et al. 2020).

This motivates us to leverage CNN in more elegant ways, i.e., 1) rather than mapping them to 2-dimensional images, we adopt one-dimensional convolution to learn the representation of the nodes; 2) rather than directly relying on the learned representation by CNN, we feed them as the input to the subsequent deep models that are equipped with more advanced architectures (e.g., Transformer in POMO) to learn deeper representations. Accordingly, in our approach, we adopt a convolutional embedding layer to identify the micro-patterns of VRP instances by exploiting the spatial invariance of the nodes. In specific, we embed the input of a node i by performing one-dimensional convolution over its K-nearest neighbors in the input graph G. Firstly,the convolutional layer computes d_h-dimensional embedding based on the coordinates of each node i and its K-nearest neighbors. Then we produce the embedding h_i for node i using a linear projection as $h_i = W_1 x_i + W_2 \overline{h_i}$, where x_i is the coordinate of node i; $\overline{h_i}$ is the convolutional results of CNN embedding layer; W_1 and W_2 are trainable matrices. With the embeddings for each node h_i which carries spatial information of its K-nearest neighbors, different deep models can process them by the encoder to further produce more informative embeddings for the decoder.

Experiments and Analysis

In this section, we conduct experiments to evaluate the efficacy of our approach on the randomly generated instances, and also the ones from two benchmark datasets, i.e., TSPLib and CVRPLib, respectively.

Experimental Settings

Typically, we consider 50 and 100 nodes for TSP and CVRP, respectively. We employ six different types of distributions to generate instances for them, including *uniform*, *explosion*, *implosion*, *expansion*, *cluster*, and *grid* (Bossek et al. 2019; Zhao et al. 2020), and also normalize them to the [0,1] square. We generate *uniform* distribution instances as the *typical* group, and the other one from the five to generate *atypical* instances as the minority group. As such, five combinations of distributions will be initially considered for each problem and size for training, while we will also additionally consider the combination of the six distributions together when applying our approach to solve the complex instances in TSPLib and CVRPLib. Particularly, regarding the *cluster* instances, we set 2 clusters for TSP50 and CVRP50, and 4 for TSP100 and CVRP100, respectively. Regarding CVRP, we compute the demand for each node as $\hat{\delta}_i = \delta_i / D$, where δ_i is uniformly sampled from $\{1, 2, \ldots, 9\}$ and D = 40 and 50 for CVRP50 and CVRP100, respectively. An additional "depot" node without demand is created in a random inside location.

We apply our approach to two different types of deep models for solving VRPs, i.e., GCN (Joshi, Laurent, and Bresson 2019) [1] and POMO (Kwon et al. 2020) [2], which are trained in the fashion of supervised learning and reinforcement learning, respectively. We term GCN and POMO that are equipped with our group DRO and CNN as *DROG* and *DROP*, respectively. Pertaining to *DROG*, we follow the suggestion stated in the original work of GCN by modifying the output to predict the next node to visit rather than an adjacency matrix. Thus, we train it as an auto-regressive model while still in the supervised way. The training labels (distance of the route) of TSP and CVRP are obtained by Concorder (Applegate et al. 2006) and LKH3 (Helsgaun 2000) respectively. We also set the inner training loop number T' in Algorithm 1 to 1, and use the cross-entropy loss to optimize edge probabilities from output layer for the sake of GCN. Pertaining to *DROP*, the original POMO introduces a data augmentation technique to exploit the symmetries of VRP instances in inference, where we adopt $\times 8$ augmentation following its original setting. We also apply group adjustments with strong ℓ_2 penalties in group DRO as did in (Sagawa et al. 2019). Regarding the convolutional embedding layer by CNN, the input length for each node equals to 2 (coordinate) for TSP and 3 (coordinate+demand) for CVRP. We extract the spatial pattern of K=10 nearest nodes with the kernel size 11 and set the number of kernels to 128. The output dimension is fixed to 128.

Our approach is implemented in PyTorch 1.2 (Paszke et al. 2019) with Python 3.7. We run the experiments on the device with a single Nvidia GeForce RTX 2080Ti GPU and a single CPU of an Intel Xeon i9-10940X CPU at 3.3 GHz. We use the SGD optimizer with minibatches and a momentum. The learning rate η is 10^{-4} for all experiments. Training time varies with the size of the problem, from a couple of hours to a week. Taking TSP100 as an example, one training epoch consumes about 11 minutes. We trained the models up to 2,000 epochs ($\sim$ 10 days) to observe full convergence, although most of them are converged within 300 epochs ($\sim$ 2 day). However, the inference time is much shorter, which is almost the same as the respective original models, as shown in the following tables. Note that all baseline deep models including GCN and POMO (and AM) are re-trained in our device, considering the instances used in this paper are essentially different from the ones in their original works.

Comparison with Baselines

We compare DROG and DROP with a variety of baselines, 1) Concorde, a specialized exact solver for TSP; 2) LKH3, a strong heuristic solver with state-of-the-art performance on many VRP variants; 3) attention model (AM) (Kool, van Hoof, and Welling 2019), the Transformer based deep model which achieves desirable results on both TSP and CVRP and recognized as a milestone; 4) GCN (original), a graph convolutional network for VRP which is trained in a supervised way and outputs the probabilities of edges that will appear in the optimal route(s) in the format of adjacency matrix; 5) POMO (original), a recent deep model developed based on AM, which is trained by RL and achieves state-of-the-art results among the deep models. Following the common settings in the works of the deep models, we use Concorde to solve TSP instances, and LKH3 to solve CVRP instances.

We train both DROG and DROP with five pairs of distribution groups in DRO. Specifically, the training set for each pair comprises 100,000 typical instances from the *uniform* distribution and 10,000 atypical instances from one of the distributions including *explosion*, *implosion*, *expansion*, *cluster*, and *grid*, respectively. The test set comprises 10,000 typical instances and 1,000 atypical instances sampled from the same pair of distribution groups used in training. Note that, the label or index of the distribution group is not required during inference. All the baseline deep models are trained and tested using the same instances as ours. Since GCN did not solve CVRP in its original work, we do not apply it to solve CVRP either.

We record the performance in terms of objective value, (optimality) gap and computation time, which are averaged over the instances in each test set. Particularly, we

[1] https://github.com/chaitjo/graph-convnet-tsp

[2] https://github.com/yd-kwon/POMO

Distribution	&Method	TSP50					TSP100					CVRP50					CVRP100				
		Obj.	Gap	Obj*	Gap*	Time	Obj.	Gap	Obj*	Gap*	Time	Obj.	Gap	Obj*	Gap*	Time	Obj.	Gap	Obj*	Gap*	Time
Explosion	Solvers	6.34	0.00%	6.69	0.00%	15m	8.58	0.00%	9.15	0.00%	1h	12.02	0.00%	12.30	0.00%	5h	17.18	0.00%	17.53	0.00%	11h
	AM	6.60	4.10%	7.11	6.28%	8m	9.01	5.01%	9.96	8.85%	31m	12.31	2.41%	12.83	4.31%	14m	17.79	3.55%	18.34	4.62%	52m
	GCN	6.47	2.05%	6.94	3.74%	25m	8.87	3.38%	9.70	6.01%	52m	-	-	-	-	-	-	-	-	-	-
	DROG	6.43	1.42%	6.75	0.90%	33m	8.74	1.86%	9.46	3.39%	1h	12.17	1.25%	12.50	1.63%	38m	**17.55**	**2.15%**	17.96	2.45%	2h
	POMO	6.52	2.84%	7.05	5.38%	19s	8.84	3.03%	9.63	5.25%	1m	12.26	2.00%	12.69	3.17%	32s	17.64	2.68%	18.14	3.48%	3m
	DROP	**6.38**	**0.63%**	**6.71**	**0.30%**	18s	**8.73**	**1.75%**	**9.40**	**2.73%**	1m	**12.15**	**1.08%**	**12.46**	**1.30%**	31s	**17.55**	**2.15%**	**17.95**	**2.40%**	3m
Implosion	Solvers	6.40	0.00%	6.74	0.00%	15m	8.76	0.00%	9.50	0.00%	1h	12.11	0.00%	12.45	0.00%	5h	17.25	0.00%	17.61	0.00%	10h
	AM	6.64	3.75%	7.23	7.27%	8m	9.12	4.11%	10.21	7.47%	33m	12.56	3.72%	12.93	3.86%	13m	17.70	2.61%	18.42	4.60%	52m
	GCN	6.50	1.56%	7.13	5.79%	25m	8.94	2.05%	9.93	4.53%	58m	-	-	-	-	-	-	-	-	-	-
	DROG	6.45	0.78%	6.79	0.74%	27m	8.83	0.80%	9.64	1.47%	1h	**12.26**	**1.24%**	**12.55**	**0.80%**	30m	17.58	1.91%	18.06	2.56%	3h
	POMO	6.57	2.66%	7.16	6.23%	22s	8.89	1.48%	9.80	3.16%	2m	12.43	2.64%	12.71	2.09%	30s	17.65	2.32%	18.25	3.63%	3m
	DROP	**6.43**	**0.47%**	**6.78**	**0.59%**	22s	**8.81**	**0.57%**	**9.58**	**0.84%**	2m	12.28	1.40%	12.60	1.20%	29s	**17.52**	**1.57%**	**18.02**	**2.33%**	3m
Expansion	Solvers	6.13	0.00%	6.19	0.00%	13m	8.19	0.00%	9.28	0.00%	1h	11.86	0.00%	12.14	0.00%	5h	16.98	0.00%	17.37	0.00%	9h
	AM	6.36	3.75%	6.55	5.82%	7m	8.46	3.30%	9.72	4.74%	28m	12.23	3.12%	12.69	4.53%	12m	17.66	4.00%	18.23	4.95%	44m
	GCN	6.24	1.79%	6.53	5.49%	23m	8.41	2.69%	9.53	2.69%	54m	-	-	-	-	-	-	-	-	-	-
	DROG	**6.18**	**0.82%**	6.26	1.13%	24m	**8.34**	**1.83%**	**9.36**	**0.86%**	1h	**11.95**	**0.76%**	**12.24**	**0.82%**	28m	17.32	2.00%	17.75	2.19%	2h
	POMO	6.29	2.61%	6.61	6.79%	19s	8.40	2.56%	9.50	2.37%	1m	12.08	1.85%	12.49	2.88%	26s	17.55	3.36%	18.04	3.86%	2m
	DROP	6.20	1.14%	**6.23**	**0.65%**	19s	**8.34**	**1.83%**	9.35	0.75%	1m	11.97	0.93%	12.28	1.15%	25s	**17.30**	**1.88%**	**17.73**	**2.07%**	2m
Cluster	Solvers	5.88	0.00%	6.17	0.00%	10m	7.44	0.00%	7.61	0.00%	1h	11.68	0.00%	11.83	0.00%	5h	16.69	0.00%	16.98	0.00%	8h
	AM	6.12	4.08%	6.67	8.10%	6m	7.81	4.97%	7.97	4.73%	26m	12.04	3.08%	12.47	5.41%	10m	17.30	3.65%	17.97	5.83%	40m
	GCN	6.02	2.38%	6.34	2.76%	22m	7.69	3.36%	7.92	4.07%	50m	-	-	-	-	-	-	-	-	-	-
	DROG	5.91	0.51%	6.24	1.13%	23m	7.64	2.69%	7.77	2.10%	1h	11.84	1.37%	12.10	2.28%	27m	17.09	2.40%	17.50	3.06%	2h
	POMO	6.07	3.23%	6.33	2.59%	18s	7.68	3.23%	7.80	2.50%	1m	11.88	1.71%	12.32	4.14%	23s	17.22	3.18%	17.73	4.42%	2m
	DROP	**5.90**	**0.34%**	**6.22**	**0.81%**	18s	**7.58**	**1.88%**	**7.73**	**1.58%**	1m	**11.79**	**0.94%**	**12.03**	**1.69%**	23s	**17.01**	**1.92%**	**17.42**	**2.59%**	2m
Grid	Solvers	6.02	0.00%	6.28	0.00%	11m	7.85	0.00%	7.99	0.00%	1h	11.72	0.00%	11.97	0.00%	5h	16.71	0.00%	17.05	0.00%	9h
	AM	6.24	3.65%	6.65	5.89%	7m	8.13	3.57%	8.51	6.51%	28m	12.14	3.58%	12.52	4.59%	10m	17.32	3.65%	17.90	4.99%	43m
	GCN	6.19	2.82%	6.38	1.59%	22m	8.09	3.06%	8.40	5.13%	51m	-	-	-	-	-	-	-	-	-	-
	DROG	**6.06**	**0.66%**	**6.30**	**0.32%**	23m	7.96	1.40%	8.15	2.00%	1h	11.83	0.94%	12.12	1.25%	27m	17.08	2.21%	17.51	2.70%	2h
	POMO	6.20	2.99%	6.41	2.07%	18s	8.01	2.04%	8.32	4.13%	1m	11.92	1.71%	12.35	3.17%	25s	17.21	2.99%	17.75	4.11%	2m
	DROP	6.10	1.33%	6.33	0.80%	17s	**7.93**	**1.02%**	**8.09**	**1.25%**	1m	**11.80**	**0.68%**	**12.08**	**0.92%**	25s	**17.00**	**1.74%**	**17.47**	**2.46%**	2m

[1] The gap is computed based on Concorder for TSP and LKH3 for CVRP respectively.
[2] $\star$ means the average results over atypical instances; **Bold** means the best result from deep models.

Table 1: Results for Comparison with Baselines

also record the objective value and gap for the atypical instances to indicate the worst-case performance. All results are summarized in Table 1. We see that our models (DROG and DROP) significantly outperform all other learning-based baselines on the five pairs of distributions, achieving much smaller objective values and gaps on both TSP and CVRP of different sizes. The superiority is more obvious for the atypical instances from the minority group, where one motivating observation is that the gaps delivered by DROP are much smaller than those by POMO, e.g., 1.25% vs 4.13% (*grid*) and 2.59% vs 4.42% (*cluster*) on TSP100 and CVRP100, respectively. We also found that while the original GCN is inferior to POMO, DROG significantly surpasses POMO for all problems and achieves even better results than DROP occasionally, e.g., TSP50 (*grid*), TSP100 (*expansion*) and CVRP50 (*implosion*).

Although the deep models like AM, GCN and POMO have demonstrated desirable performance when they are trained and tested solely using the *uniform* distribution (as reported in their original works), their overall and worst-case performance obviously deteriorate in the presence of atypical instances (as shown in Table 1). One of the major reasons is that the models would be overwhelmed by the typical instances (majority) even if they also consider the atypical instances (minority) in training. In contrast, our approach elegantly balances the training on different distribution groups (i.e. either the majority or minority one) by leveraging the group DRO, and enhance the distribution-aware embedding by CNN. Moreover, our approach is versatile to different deep models either trained by supervised or reinforcement learning, such as GCN and POMO used in our experiments. Besides, our approach would not incur much additional time for inference, as suggested in Table 1.

Ablation Study

To further verify the effectiveness of the components in our approach, i.e., group DRO and CNN, we conduct an ablation study on TSP100 by separately removing the corresponding component from our DROP. The experimental setting is similar to the previous one, and the results are displayed in Table 2. When POMO is solely equipped with CNN, we observe that the average objective values for the overall instances and the atypical instances are both decreased. Benefiting from the initial distribution-aware embedding, it indicates that the convolutional embedding layer can help learn better spatial patterns of nodes and improve the gen-

	Concorde				POMO				POMO+CNN				POMO+DRO				DROP			
Distribution	Obj.	Gap	Obj*	Gap*	Obj.	Gap	Obj*	Gap*	Obj.	Gap	Obj*	Gap*	Obj.	Gap	Obj*	Gap*	Obj.	Gap	Obj*	Gap*
Explosion	8.58	0.00%	9.15	0.00%	8.84	3.03%	9.63	5.25%	8.80	2.56%	9.62	5.14%	8.76	2.10%	9.45	3.28%	**8.73**	**1.75%**	**9.40**	**2.73%**
Implosion	8.76	0.00%	9.50	0.00%	8.89	1.48%	9.80	3.16%	8.83	0.80%	9.76	2.74%	8.83	0.80%	9.59	0.95%	**8.81**	**0.57%**	**9.58**	**0.84%**
Expansion	8.19	0.00%	9.28	0.00%	8.40	2.56%	9.50	2.37%	8.37	2.20%	9.45	1.83%	8.35	1.95%	9.37	0.97%	**8.34**	**1.83%**	**9.35**	**0.75%**
Cluster	7.44	0.00%	7.61	0.00%	7.68	3.23%	7.80	2.50%	7.61	2.28%	7.79	2.37%	7.62	2.42%	7.75	1.84%	**7.58**	**1.88%**	**7.73**	**1.58%**
Grid	7.85	0.00%	7.99	0.00%	8.01	2.04%	8.32	4.13%	7.97	1.53%	8.27	3.50%	7.95	1.27%	8.12	1.63%	**7.93**	**1.02%**	**8.09**	**1.25%**

Table 2: Results for Ablation Study

Instance	Opt.	AM	POMO	DROG	DROP
Eil51	426	439	436	427	**426**
Berlin52	7542	8352	7836	7553	**7544**
St70	675	691	683	684	**679**
rat99	1211	1340	1286	**1233**	1248
KroA100	21282	46621	38452	25196	**24623**
KroB100	22141	37921	33521	26583	**24874**
KroC100	20749	34258	30736	**24343**	24785
KroD100	21294	36141	29512	23633	**23257**
KroE100	22068	29628	26829	26289	**26057**
rd100	7910	8252	8180	8137	**8043**
lin105	14379	15148	14922	14876	**14688**
pr107	44303	53846	52846	**46572**	47853
ch150	6528	6930	6844	6792	**6709**
rat195	2323	2612	2554	2466	**2403**
kroA200	29368	35637	34972	34682	**34275**
X-n101-k25	27591	38264	29484	29167	**28949**
X-n106-k14	26362	27923	27762	27331	**27308**
X-n110-k13	14971	16320	15896	**15226**	15386
X-n115-k10	12747	14055	13952	13921	**13783**
X-n120-k6	13332	14456	14351	14227	**14058**
X-n125-k30	55539	74329	69560	64596	**61382**
X-n129-k18	28940	30869	30155	30181	**30075**
X-n134-k13	10916	13952	13483	13117	**12846**
X-n139-k10	13590	14893	14132	14153	**13979**
X-n143-k7	15700	18251	17923	**17547**	17682
X-n153-k22	21220	38423	26386	24591	**24386**
X-n157-k13	16876	22051	19978	18993	**18378**
X-n181-k23	25569	27826	27428	27232	**27094**
X-n190-k8	16980	37820	22310	20682	**19864**
X-n200-k36	58578	76528	73135	68283	**64921**
Avg. Gap	0.00%	29.93%	17.57%	9.68%	**8.08%**

[1] **Bold** means the best result from deep models.

Table 3: Generalization Results on TSPLib and CVRPLib

eralization performance. On the other hand, when POMO is solely equipped with group DRO, it can also ameliorate the performance over the original POMO on both the overall instances and the atypical instances. We also found that the advantage brought by DRO (POMO+DRO) is more significant than that of CNN (POMO+CNN), especially on the atypical instances, since it is specialized to improve the performance for instances of the worst-case group. Finally, when equipped with both CNN and group DRO, our DROP achieves further better results than the respective variants and the original model, which well justified the effectiveness of the two components that could jointly promote the performance on both the overall and atypical instances.

Evaluation on Benchmark Dataset

We continue to evaluate DROG and DROP on the public benchmark dataset, i.e., TSPLib (Reinelt 1991) and CVRPLib (Uchoa et al. 2017), whose instances may follow various distributions that are totally different from ours. We train our models on 150,000 instances, which include 100,000 instances from *uniform* distribution, and 50,000 instances from the other five, with 10,000 for each (although they do not need to be the same). Then we apply our models trained on TSP100 and CVRP100 to solve some representative instances, whose sizes range from 51 to 200. We separately display results for TSP and CVRP in the upper and lower half of Table 3. The baseline models are also re-trained using the same instances. It is revealed that DROG and DROP achieve near-optimal solutions on both TSP and CVRP. Both DROG and DROP, with average gaps of 9.68% and 8.08%, produce much better results than that of AM and POMO, with average gaps of 29.93% and 17.57%, respectively, although they are using the same training set as ours. This superiority suggests that, our approach equipped with group DRO and CNN, allows the deep models to favorably generalize to different distributions and sizes. Last but not least, the inference of DROG and DROP are almost as efficient as AM and POMO, respectively, indicating that our approach introduces negligible additional computation overhead.

Conclusion

In this paper, we exploit group DRO (distributionally robust optimization) to enhance the cross-distribution generalization ability for deep models that are used to solve VRPs. We also leverage an elegant CNN to learn the initial distribution-aware representations of VRP instances. Our approach is readily applicable with either supervised or reinforcement learning and evaluated with two well-known deep models, i.e., GCN and POMO. Empirical results show that our approach significantly improves the cross-distribution generalization performance and outstrips other learning based methods on both the synthesized and benchmark dataset. The ablation study also verifies the efficacy of components in our approach. In future, we will investigate more flexible combinations of distributions and tackle problems of larger scales.

Acknowledgments

This study is supported under the RIE2020 IAF-ICP Funding Initiative and Singapore Telecommunications Limited (Singtel) through SCALE@NTU. It is also partially supported by the National Natural Science Foundation of China under Grant 61803104.

References

Applegate, D.; Bixby, R.; Chvatal, V.; and Cook, W. 2006. Concorde TSP solver. *URL http://www.math.uwaterloo.ca/tsp/concorde*.

Bello, I.; and Pham, H. 2017. Neural Combinatorial Optimization with Reinforcement Learning. In *the 5th International Conference on Learning Representations*.

Ben-Tal, A.; Den Hertog, D.; De Waegenaere, A.; Melenberg, B.; and Rennen, G. 2013. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2): 341–357.

Bengio, Y.; Lodi, A.; and Prouvost, A. 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421.

Bossek, J.; Kerschke, P.; Neumann, A.; Wagner, M.; Neumann, F.; and Trautmann, H. 2019. Evolving diverse TSP instances by means of novel and creative mutation operators. In *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, 58–71.

Chen, X.; and Tian, Y. 2019. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, 6278–6289.

Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; and Schulman, J. 2019. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, 1282–1289.

Dantzig, G. B.; and Ramser, J. H. 1959. The truck dispatching problem. *Management science*, 6(1): 80–91.

Duchi, J. C.; Hashimoto, T.; and Namkoong, H. 2019. Distributionally robust losses against mixture covariate shifts. *arXiv preprint arXiv:2007.13982*.

Han, H.; and Ponce Cueto, E. 2015. Waste collection vehicle routing problem: literature review. *PROMET-Traffic&Transportation*, 27(4): 345–358.

Hashimoto, T.; Srivastava, M.; Namkoong, H.; and Liang, P. 2018. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*, 1929–1938.

Helsgaun, K. 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1): 106–130.

Hottung, A.; and Tierney, K. 2020. Neural large neighborhood search for the capacitated vehicle routing problem. In *Proceedings of the 24th European Conference on Artificial Intelligence*.

Hu, W.; Niu, G.; Sato, I.; and Sugiyama, M. 2018. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, 2029–2037.

Joshi, C. K.; Laurent, T.; and Bresson, X. 2019. An efficient graph convolutional network technique for the travelling salesman problem. In *INFORMS Annual Meeting*.

Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; and Song, L. 2017. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, 6348–6358.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! In *the 7th International Conference on Learning Representations*.

Kuhn, D.; Esfahani, P. M.; Nguyen, V. A.; and Shafieezadeh-Abadeh, S. 2019. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In *Operations Research & Management Science in the Age of Analytics*, 130–166. INFORMS.

Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; and Min, S. 2020. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. In *Advances in Neural Information Processing Systems*.

Li, J.; Ma, Y.; Gao, R.; Cao, Z.; Andrew, L.; Song, W.; and Zhang, J. 2021. Deep Reinforcement Learning for Solving the Heterogeneous Capacitated Vehicle Routing Problem. *IEEE Transactions on Cybernetics*.

Ling, Z.; Tao, X.; Zhang, Y.; and Chen, X. 2020. Solving Optimization Problems Through Fully Convolutional Networks: An Application to the Traveling Salesman Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–11.

Lu, H.; Zhang, X.; and Yang, S. 2019. A learning-based iterative method for solving vehicle routing problems. In *International Conference on Learning Representations*.

Ma, Y.; Li, J.; Cao, Z.; Song, W.; Zhang, L.; Chen, Z.; and Tang, J. 2021. Learning to Iteratively Solve Routing Problems with Dual-Aspect Collaborative Transformer. In *Advances in Neural Information Processing Systems*, volume 34.

Malheiros, I.; Ramalho, R.; Passeti, B.; Bulhões, T.; and Subramanian, A. 2021. A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem. *Computers & Operations Research*, 129: 105196.

Miki, S.; and Ebara, H. 2019. Solving Traveling Salesman Problem with Image-Based Classification. In *Proceedings of the 31st International Conference on Tools with Artificial Intelligence*, 1118–1123.

Namkoong, H.; and Duchi, J. C. 2017. Variance-based regularization with convex objectives. In *Advances in Neural Information Processing Systems*, 2975–2984.

Oren, Y.; Sagawa, S.; Hashimoto, T.; and Liang, P. 2019. Distributionally Robust Language Modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 4227–4237.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.;

and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035.

Reinelt, G. 1991. TSPLIB—A traveling salesman problem library. *ORSA journal on computing*, 3(4): 376–384.

Sagawa, S.; Koh, P. W.; Hashimoto, T. B.; and Liang, P. 2019. Distributionally robust neural networks. In *International Conference on Learning Representations*.

Sinha, A.; Namkoong, H.; and Duchi, J. 2018. Certifying Some Distributional Robustness with Principled Adversarial Training. In *International Conference on Learning Representations*.

Steever, Z.; Karwan, M.; and Murray, C. 2019. Dynamic courier routing for a food delivery service. *Computers & Operations Research*, 107: 173–188.

Sun, Y.; Wang, X.; Liu, Z.; Miller, J.; Efros, A.; and Hardt, M. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, 9229–9248.

Uchoa, E.; Pecin, D.; Pessoa, A.; Poggi, M.; Vidal, T.; and Subramanian, A. 2017. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3): 845–858.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, 2692–2700.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4): 229–256.

Wu, Y.; Song, W.; Cao, Z.; Zhang, J.; and Lim, A. 2021. Learning Improvement Heuristics for Solving Routing Problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021a. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of 35th AAAI Conference on Artificial Intelligence*.

Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021b. NeuroLKH: Combining Deep Learning Model with Lin-Kernighan-Helsgaun Heuristic for Solving the Traveling Salesman Problem. In *Advances in Neural Information Processing Systems*, volume 34.

Zhao, K.; Liu, S.; Rong, Y.; and Yu, J. X. 2020. Leveraging TSP Solver Complementarity via Deep Learning. *arXiv e-prints*, arXiv–2006.