Learning Generalizable Heuristics for Solving Vehicle Routing Problem under Distribution Shift

Jiang Yuan
School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

2024

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

05/08/2022

Date

Jiang Yuan

Jiang Yuan

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

05/08/2023

.....

Date

.....

Prof. Zhang Jie

Authorship Attribution Statement

This thesis contains material from 3 paper(s) published in the following peer-reviewed conferences in which I am listed as an author.

Chapter 3 is published as **Yuan Jiang**, Yaoxin Wu, Zhiguang Cao, Jie Zhang. Learning to Solve Routing Problems via Distributionally Robust Optimization. 36th AAAI Conference on Artificial Intelligence, 2022. The contributions of the co-authors are as follows:

- I proposed the method, conducted experiments and wrote the manuscript draft. The other authors gave me valuable advice at every stage.
- Prof. Zhang, Yaoxin Wu and Zhiguang Cao suggested the project direction and polished the manuscript.

Chapter 4 is published as **Yuan Jiang**, Zhiguang Cao, Yaoxin Wu, Jie Zhang. Multi-View Graph Contrastive Learning for Solving Vehicle Routing Problems. 39th Conference on Uncertainty in Artificial Intelligence, 2023. The contributions of the co-authors are as follows:

- I proposed the method, conducted experiments and wrote the manuscript draft. Prof. Zhang, Yaoxin Wu and Zhiguang Cao gave me valuable advice for preparing the manuscript. Yaoxin Wu and Zhiguang Cao helped polish the manuscript.

Chapter 5 is submitted to **Yuan Jiang**, Zhiguang Cao, Yaoxin Wu, Wen Song, Jie Zhang. Ensemble-based Deep Reinforcement Learning for Vehicle Routing Problems under Distribution Shift. 37th International Conference on Advances in Neural Information Processing Systems, 2023. The contributions of the co-authors are as follows:

- I proposed the method, conducted experiments and wrote the manuscript draft. Prof. Zhang, Zhiguang Cao, Wen Song provided me with valuable comments to prepare the manuscript. Zhiguang Cao, Wen Song and Yaoxin Wu helped polish the manuscript.

..... 05/08/2022 .....

Date

Jiang Yuan

Jiang Yuan

Acknowledgements

I would like to thank my professor, Zhang Jie, for his guidance, supervision and support throughout my PhD studies. He has given me valuable advice, constructive feedback and insightful suggestions on my research. He has also encouraged me to pursue my interests, explore new directions and challenge myself. I am grateful for taking care of me and helping me grow as a researcher. He has been a great example and a caring supervisor.

I would also like to thank Cao Zhiguang, for his guidance and encouragement. He has guided me to purchase my Ph.D. degree and helped me a lot in figuring out ideas and polishing my manuscripts. He has also shared his expertise, experience and resources with me. He is a great mentor and a kind person. I greatly thank Wu Yaoxin, Song Wen and Ewa Andrejczuk. They have provided me with valuable feedback, insights and ideas on our research topics. They have also supported me in various aspects of our research. I am thankful for their cooperation and friendship.

I am grateful to the team members who generously shared their knowledge and guidance with me on my research. Thanks to Li Xiaoming, Chen shuo, Rubha Shri Narayanan, Avataram Venkatavaradan Prituja, Xin Liang, Zhang Xianli, Zhou Jianan and Xia Yunwen for their support and assistance in my study, research and project.

Moreover, I deeply appreciate Vivienne Liao Weiwen, Yang Fushi and Zhang Jia for their companionship and encouragement during my hard times, and thank my family for supporting my life and study since childhood.

I would like to express my deepest gratitude to my fiancee, Qingyang, for her constant love, encouragement and support throughout my PhD journey. She has been my source of inspiration, motivation and happiness. She has always believed in me and my research, even when I faced challenges and doubts. She has also made me a better person.

Abstract

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) 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, dial-a-ride and courier delivery. Studies on VRP with deep (reinforcement) learning have been emerging in recent years. Different from the conventional methods, this line of work 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. Towards reducing the gaps to the highly optimized conventional heuristic solvers, a large number of efforts have been made to invent various deep models to solve the VRP variants, i.e., traveling salesman problem (TSP) and capacitated vehicle routing problem (CVRP). 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. 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 in the group of the atypical instances.

To address this issue, in the first work, 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 convolutional 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.

Secondly, motivated by the facts that, 1) a VRP instance can always be represented as a graph, 2) the VRP solution depends on the pattern of the graph (e.g. the distribution of nodes), we postulate that transferable structural patterns across diverse graphs could be helpful to improve the generalization against different distributions. Especially, similar local patterns may distribute across graphs even if those graphs belong to different distributions. On the other hand, recent advances in computer vision (CV) and natural language processing (NLP) have testified that pre-training an encoder network in a contrastive learning manner can produce more informative and transferable representations for downstream tasks. With the above principle, in my second work, we propose a multi-view graph contrastive learning (MVGCL) approach to foster the generalization capability of neural heuristics for VRPs, by mining the underlying patterns across graphs.

In the third work, we propose an ensemble-based deep reinforcement learning method for VRPs, which learns a group of diverse sub-policies to cope with various instance distributions. In particular, to prevent convergence of the parameters to the same one, we enforce diversity across sub-policies by leveraging Bootstrap with random initialization. Moreover, we also explicitly pursue inequality between sub-policies by exploiting regularization terms during training to further enhance diversity. Experimental results show that our methods are able to outperform the state-of-the-art neural baselines on randomly generated instances of various distributions, and also generalizes favourably on the benchmark instances from TSPLib and CVRPLib, which confirmed the effectiveness of the whole method and the respective designs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The vehicle routing problem (VRP) is a type of combinatorial optimization problems (COPs) with significant real-world applications, especially in the field of logistics [1]. The goal is to optimally route a fleet of vehicles, originating at a depot, to serve a set of clients. There is no known polynomial-time algorithm to solve it or verify VRPs' solution, and the required solution time increases exorbitantly with size [2]. Given the NP-hard nature, heuristic methods are usually preferred to solve VRPs in practice. However, traditional heuristics always treat each VRP instance independently, resulting in limited computational efficiency or solution quality [3]. Hence, there is growing interest in developing neural heuristics based on deep learning to exploit the underlying patterns across VRP instances and then generalize to unseen ones [4]. Recently, there is a growing trend towards applying deep reinforcement learning (DRL) to solve combinatorial optimization problems, especially the routing problems [5–12]. On the one hand, DRL adopts advanced deep architectures (e.g., encoder-decoder structure) as the policy network to automatically learn the decision rules by exploiting the underlying pattern among a large number of instances. On the other hand, DRL does not need the optimal solutions as the ground-truth labels, which are computationally expensive to be acquired. As such, it achieves much success in solving the combinatorial optimization problems.

Early neural heuristics for VRPs were primarily supervised [13, 14], of which the performance depends heavily on the ground-truth labels, making them less desirable due to the expensive computation for obtaining optimal solutions. By contrast, deep reinforcement learning (DRL) relies on feedback from the environment rather than the ground-truth optimal solution as a reward, allowing neural heuristics to perform favorably in solving VRPs [15]. Typically, DRL can be used to train networks for selecting the next client (or node) to visit in VRPs [6, 16–18], which is more capable of generalizing to different problem sizes in comparison with the supervised counterparts [19].

1.2 Motivations

While DRL-based neural heuristics have achieved a series of successes for VRPs, most of them perform reasonably well only when the underlying deep models are trained and tested on the independent and identically distributed (i.i.d.) instances with respect to the client locations, e.g., the ones generated using the uniform distribution. In the presence of distribution shifts, they struggle to deliver desirable performance, which greatly hinders the applications of neural heuristics in practice. Especially in real-world VRP instances, the client locations may vary dramatically according to weekday or weekend, sunny or rainy weather, etc., leading to various or even unknown distributions. Therefore, how to strengthen the cross-distribution generalization capability is of great importance to the neural heuristics for VRPs. Although several preliminary studies have investigated this issue, including the curriculum-learning-based hardness-adaptive generator (HAC, [20]) and adaptive multi-distribution knowledge distillation (AMDKD, [20]), they still fall short of optimality due to their inferior performance or less practical settings. HAC focuses on the Traveling Salesman Problem (TSP) but is limited to instances with up to 50 nodes in their setting. Its heavy reliance on uniform and Gaussian distributions hampers its generalization performance when faced with other distribution types. On the other hand, AMDKD demands a substantial amount of time for training both the teacher model and the student model. Its performance heavily depends on pre-specified distributions, which is impractical to obtain in many real-world applications. In summary, while these approaches contribute valuable insights, there is still room for improvement in addressing the challenges posed by distribution

shifts in optimization problems. Hence, it remains challenging to comprehensively and effectively expand the limited generalization capability of the neural heuristics against the distribution shifts for VRPs.

To address these issues, this thesis focuses on developing neural heuristics to solve routing problems under distribution shifts. Specifically, this thesis targets VRPs which are mostly studied in the literature of learning to optimize [16, 21]. The research on VRPs is promising to boost the development of neural heuristics for similar combinatorial optimization problems, e.g., 3D bin packing and resource/-task scheduling [22–24]. Existing neural-based methods mostly assume the distributions are the same during both training and testing, which might limit their generalization performance. In contrast, we propose three neural heuristics that learn to generalize to different unseen distributions: group distributionally robust optimization (group DRO), multi-view graph contrastive learning (MVGCL) and ensemble-based deep reinforcement learning (EL-DRL).

We first propose a method that uses *group distributionally robust optimization* (group DRO) to train deep models on different groups of instances (more than one group), where each group has its own distribution. Our method tries to minimize the loss for the *worst-case* group during training, by alternating between optimizing the weights for each group and the parameters for the deep model. Our method does not require labeling the distribution of each group during testing. Moreover, we use convolutional neural network (CNN) to learn initial representations of VRP instances that can capture the distribution-aware features in spatial patterns and improve the performance further. Our experiments show that our method can achieve better performances on both the overall instances and the atypical instances, and can be applied to different deep models. We do not aim to beat the state-of-the-art deep methods for solving VRPs in all aspects, but to make them more robust and generalize better across distributions.

However, collecting data with labeled classes of distribution is challenging for many applications. Therefore, we developed an unsupervised learning method that automatically explores and exploits the distribution of instances. We call this method multi-view graph contrastive learning (MVGCL). From the perspective of spatial distribution of the routing graph, local structure and global distribution also have been considered in our second approach. We assume that transferable structural

patterns across diverse graphs can help neural heuristics for VRPs generalize better across distributions, since VRP solutions depend on the graph patterns. We also observe that similar local patterns can exist across graphs of different distributions. Moreover, contrastive learning has been shown to produce more informative and transferable representations for downstream tasks in CV and NLP. Based on these, we propose a multi-view graph contrastive learning (MVGCL) approach that uses contrastive learning with a weighted random walk augmentation and a distribution-preserved augmentation to mine the local and global patterns across graphs of VRP instances of various distributions.

In MVGCL, we leverage both local and global information to enhance the generalization performance. We acknowledge that utilizing different channels can extract various kinds of information, but a single model may have limited ability to learn from multiple channels. Therefore, inspired by the fact that an ensemble of varied models can leverage their respective strengths to collaboratively handle different distributions, we propose an end-to-end ensemble-based deep reinforcement learning (EL-DRL) to solve VRPs in the presence of distribution shifts, which ensures diverse and complementary learning across different sub-models (or sub-policies), thereby fostering the cross-distribution generalization capability. Specifically, we first extend the REINFORCE algorithm and policy gradient in DRL to the ensemble setting by adaptively training a set of sub-policies on instances of mixed distributions to collaboratively solve a VRP instance. Then, to encourage diversity among the sub-policies, we harness Bootstrap with random initialization to enforce that each sub-policy is updated according to different loss signals. Additionally, we also exploit different regularization terms to prevent the collapse of sub-policies by explicitly pursuing inequality among them.

We design our neural heuristics with minimal expertise and tuning, and the results show their superior generalization performance to different distributions. We believe that these approaches can unleash the potential of deep reinforcement learning and reduce the gap to strong and optimized solvers under distribution shifts. Moreover, the ideas in this thesis can be applied to solve other similar COPs beyond routing problems.

1.3 Main Contributions

The methods of this thesis employ deep reinforcement learning (DRL) to train and learn generalizable neural heuristics for solving routing problems. To improve the performance on atypical distributions, we propose a method that trains deep models on different groups of VRP instances using group DRO, which minimizes the worst-case group loss. From the perspective of spatial structure in routing graphs, we propose MVGCL, which uses contrastive learning with two augmentations to learn transferable graph patterns for VRPs. Also, we propose EL-DRL to train an ensemble of sub-policies with maximized diversity for solving different VRPs instances. The major contributions of the proposed methods are summarized in the following:

- **Learning to Solve Routing Problems via Distributionally Robust Optimization.** 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 fashion of either supervised or reinforcement learning, e.g., GCN [14] and POMO [17], 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.

- **Learning to Solve Routing Problems Via Multi-View Graph Contrastive Learning.** We propose a multi-view graph contrastive learning (MVGCL) approach to foster the generalization capability of neural heuristics for VRPs, through mining the underlying patterns across graphs. Specifically, given a collection of graphs of VRP instances of various distributions, our MVGCL exploits contrastive learning with a weighted random walk augmentation to identify the local transferable patterns, and a distribution-preserved augmentation to identify the global distribution across these graphs. This pre-trained graph neural network (GNN) acts as the encoder, which learns the representation in two views,

1) a *node* embedding with respect to the local structural similarity; 2) a *graph* embedding with respect to the overall distribution information. Subsequently, the two learnt embeddings are employed to facilitate a neural heuristic (i.e. POMO [17]) and its active search scheme [25] (in the inference phase) for downstream route construction, respectively. In this way, our approach not only learns the transferable pattern across various distributions, but also adjusts itself with individual instances. We conduct extensive experiments on two widely studied VRP variants, i.e., TSP and CVRP. Results on randomly generated instances and benchmark ones verified its effectiveness.

- **Learning to Solve Routing Problems Via Ensemble-based Deep Reinforcement Learning.** We empirically demonstrate that a simple ensemble of similar sub-policies leads to limited performance in solving VRPs with distribution shifts. We propose an ensemble-based deep reinforcement learning (EL-DRL), and its ensemble-based policy gradient allows multiple sub-policies to learn to collaboratively leverage their strengths on respective distributions. We strengthen diversity by assigning different loss signals to sub-policies and applying regularization based on inequality metrics. Our EL-DRL adaptively trains sub-policies on instances of various distributions and is flexible without manually specifying instance class or group [20, 26] to train each sub-policy. We deploy POMO [17] as the sub-model in EL-DRL for solving the TSP and CVRP on both synthetic and benchmark instances of various distributions. The results verified the superior cross-distribution generalization of our EL-DRL.

1.4 Outline of the Thesis

The structure of this thesis is organized in the following sequence. In Chapter 2, we review the existing neural heuristic approaches for vehicle routing problems and introduce the recent progress of distributionally robust optimization, graph contrastive learning and ensemble learning. In Chapter 3, we propose a group DRO approach to enhance the worst-case generalization of deep models, and show the experimental results. In Chapter 4, we develop a framework for incorporating exact node-level and graph-level information to help deep models solve instances from different distributions, and present the experimental results. In Chapter 5,

we design an ensemble method to learn diverse sub-models for tackling VRPs under distribution shifts, and report the experimental findings. Finally, we make conclusions for this thesis and discuss future work and limitations in Chapter 6.

Chapter 2

Literature Review

For this chapter, we start with the review of deep learning methodologies in addressing the intricacies of Vehicle Routing Problems (VRPs) and some other Combinatorial Optimization Problems (COPs). Our exploration underscores the capacity of deep models to automate heuristic solutions for COPs, showcasing their distinct advantages over traditional methodologies. An examination of diverse deep learning approaches for VRPs, including Pointer Networks, Attention Networks, and Graph Convolutional Networks, is presented. The challenge of extending these models beyond their original training distributions is also discussed. To explore possible solutions to mitigate this challenge, we elaborate Distributionally Robust Optimization (DRO) as an approach to bolster generalization, with specific emphasis on the efficacy of group DRO in enhancing cross-distribution performance. Our analysis also extends to the literature on Graph Contrastive Learning (GCL), underlining the necessity for tailoring augmentation techniques for VRP graphs. Finally, we draw attention to the potential prowess of Deep Ensemble Learning, showcasing its role in encapsulating diverse patterns within complex problems.

2.1 Deep Models for VRPs

Deep learning has been widely applied to various COPs in recent years, such as routing [27–29], scheduling [30–32], knapsacking [33–35] and bin packing [24, 36, 37]. Other specific COPs, such as maximum cut, boolean satisfiability, graph coloring, are also covered by survey papers [3, 4, 38]. Vehicle routing problems

are a major class of COPs that have many variants and attract more attention from both academia and industry. 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 [13]. Similarly, Joshi et al. (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 [6, 10, 16]. Specifically, 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 [9, 39–41]. 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. Differently, a concurrent line of *improvement* methods [9, 18, 39, 40, 42] emphasize improving an initial but complete solution via iterative local operations. Among them, Kim et al. [18] propose to learn collaborative policies (LCP) with a seeder to explore large solution space and a reviser to improve the solution for local segments. On the other hand, Graph Neural Networks or its variants such as graph convolutional networks (GCNs) and graph attention networks (GATs) are also exploited on VRP graphs [14, 22, 43]. Besides, Li et al. [44] propose guided tree search by leveraging GCN embeddings. Fu et al. [45] use a graph convolutional residual network and a Monte Carlo tree to generalize to larger instances on TSP.

The aforementioned neural methods have exhibited impressive performance when the training and testing instances share the same distribution regarding the node locations, e.g., uniform [17, 46]. However, Geisler et al. [47] and Zhang et al. [48]

show that simply applying those neural heuristics to other distributions may cause considerably inferior solutions. To generalize the neural heuristics beyond the single (uniform) distribution used in training, Zhang et al. [48] propose a curriculum learning-based AM (HAC) trained on instances of different hardness. HAC focuses on the Traveling Salesman Problem (TSP) but is limited to instances with up to 50 nodes in their setting. Its heavy reliance on uniform and Gaussian distributions hampers its generalization performance when faced with other distribution types. Those instances are generated by a hardness-adaptive generator with mixed-Gaussian distribution, which limits its generalization to more others. A recent study (AMDKD, [20]) distills the knowledge from various pre-trained teacher models, which manually assigns specific distribution to train each teacher model. On the other hand, AMDKD demands a substantial amount of time for training both the teacher model and the student model. Its performance heavily depends on pre-specified distributions, which is impractical to obtain in many real-world applications. In summary, while these approaches contribute valuable insights, there is still room for improvement in addressing the challenges posed by distribution shifts in optimization problems.

2.2 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 [49, 50]. This issue about the generalization in out-of-distribution is still challenging for the general learning-based approaches, either in supervised or reinforcement learning [51, 52].

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 [53, 54]. It was originally designed for classic under-parameterized models to reduce the training loss, by regularizing the models and defending them against adversarial examples [55, 56]. 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 [49, 57]. In this line, Sagawa et al. (2019) applied group DRO in the

over-parameterized 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 chapter, 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.

2.3 Graph contrastive learning

In addition to DRO, Graph Contrastive Learning emerges as a promising avenue with the potential to enhance the overall generalization performance on routing graphs. Contrastive learning (CL) is a type of self-supervised learning. It is usually used to identify the similarity among the unlabeled data and learn inherent representations across instances, which has been widely explored in CV [59–61] and NLP [62, 63].

To tackle the geo-graphic data, graph contrastive learning (GCL) has been proposed [64–66], and a series of augmentation techniques to generate contrastive samples based on the original graph have also been accordingly developed [67–69], such as attribute removing, edge adding/masking, and subgraph/graph diffusion. Among them, Qiu et al. [64] define the contrastive samples as the r-ego sub-network of the input nodes and then apply the pre-trained GNN on tasks of node or graph classification. Hassani and Khasahmadi [65] contrastively learn embeddings from the first-order neighbours and graph diffusion. GraphCL [70] handpicks ad-hoc augmentations (node dropping, edge perturbation, attribute masking and subgraph sampling) to provide specific contrastive samples for graph-level representation learning, while this augmentation selection is made automated in its subsequent work [67]. Similarly, Yin et al. [68] propose adaptive augmentation to remove edges.

However, the augmentation methods of existing graph contrastive learning are not directly applicable to our routing tasks in that, 1) nodes in VRPs like TSP only has coordinates as the attribute (unlike social networks with rich attributes such as age, gender and country of a person), which makes it harder to distinguish the node from each other by attribute masking [70]; 2) VRP graphs are fully-connected,

augmentations like node dropping or edge perturbation [66, 71] may violate the connectedness. Worse still, the original graph distribution could be distorted by altering its structure. Although recent works [68, 72] attempt to preserve graph class labels during augmentation, it is impractical to acquire all distribution labels for VRPs. Furthermore, unlike traditional network embeddings [73–75] or recent works that pre-train GNNs with attributed graphs and then directly apply them to instances of the same domain [76], our goal is to pre-train a GNN for learning local structure across distributions and global graph embedding of the instance, which allows the neural heuristic to solve VRPs of various distributions effectively.

2.4 Deep Ensemble Learning

Despite the increased capacity compared to a single model, practical studies often use multiple models to improve the generalization ability. Deep ensemble learning models combine the strengths of both deep learning and ensemble learning to achieve superior generalization performance [77, 78]. Ensemble learning is recognized as a promising solution for adaptation to various tasks [77–79], due to the unique advantage that respective sub-models hold in solving different tasks. Among of them, Q-ensemble and policy ensemble methods [80] are promising ones. Q-ensemble focuses on uncertainty-based offline reinforcement learning. It leverages the confidence of Q-value predictions without requiring data distribution estimation or sampling. By penalizing out-of-distribution data points with high prediction uncertainties, Q-ensemble achieves substantial improvements over existing offline RL methods [81]. Adapting ensemble methods to policy learning remains an open problem, which combines sub-policy training and policy ensemble under the same reinforcement learning objective. These methods [82] optimizes both ensemble diversity and individual accuracies, leading to robust policies. A few works have (implicitly) borrowed ideas from ensemble learning to solve VRPs [10, 40]. For example, MDAM [10] leverages the AM equipped with multiple decoders to solve VRPs, and each decoder corresponds to a sub-model in ensemble learning. However, due to the demanding computation, no mechanism is in place to ensure the diversity of constructed solutions beyond the first node. L2I [40] requires manually assigning modification rules to train different policies separately, which also takes prohibitively long runtime to search for better solutions iteratively. In summary,

these works focus more on combining multiple models rather than promoting the automatic acquisition of diverse knowledge. On the other hand, diversity is crucial in ensemble learning for improving prediction performance and robustness [83, 84]. Ensemble diversity measures, algorithms for creating diverse ensembles, and ensemble consensus have been investigated to produce high-accuracy ensemble outputs [85, 86]. Our work differs from existing DRL-based methods for VRPs in that we explicitly consider and enforce diversity among sub-models, allowing them to capture different patterns from various distributions without much manual interference.

Chapter 3

Learning to Solve Routing Problems via Distributionally Robust Optimization

Combinatorial optimization problems (COPs), known for their NP-hardness and intricate discrete search spaces, have diverse applications. The Vehicle Routing Problem (VRP) stands out in logistics, optimizing delivery routes for geographically scattered customers. Recent years have seen the emergence of deep (reinforcement) learning in VRP studies, aiming to autonomously derive heuristic policies through neural networks. Efforts to bridge gaps with optimized heuristic solvers have led to various deep models tailored to VRP variants such as the Traveling Salesman Problem (TSP) and Capacitated Vehicle Routing Problem (CVRP).

Despite successes, existing deep models train under a uniform distribution assumption, limiting cross-distribution generalization performance. Atypical instances that deviate from training norms can result in inferior performance. To address this, in this chapter, we propose an approach based on group Distributionally Robust Optimization (group DRO) to train deep models across diverse distribution-defined groups. This approach iteratively optimizes instance group weights and deep model parameters, eliminating distribution labeling during inference. We also use a convolutional neural network (CNN) to identify initial spatial patterns in VRP instances, enhancing distribution-aware feature learning and solution quality. Empirical evaluation confirms the approach's efficacy in enhancing overall and

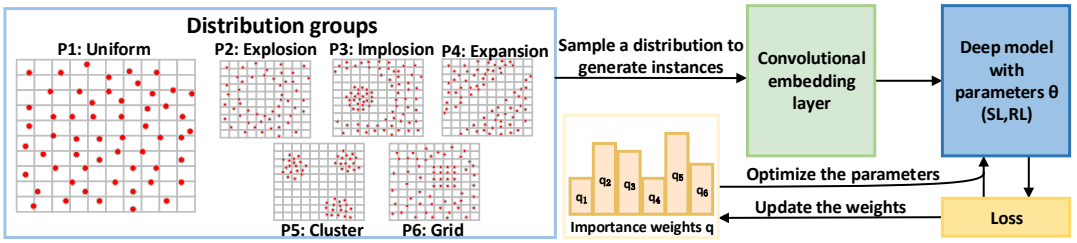

FIGURE 3.1: An example illustration of our approach with VRP instances sampled from six distribution groups, which alternately optimizes the parameters θ of a deep model and updates the importance weights q of the distribution groups in the training set.

worst-case performance across deep models and benchmark datasets. While not claiming supremacy in a single uniform distribution, this approach bolsters deep models' cross-distribution generalization for better VRP solving. The advantages of this method include adding robustness through group DRO, adaptable to diverse deep learning paradigms, and using CNN for enhanced feature representation.

3.1 Preliminaries

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

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

3.1.2 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,

$$\hat{\theta}_{\text{ERM}} := \arg\min_{\theta \in \Theta} \mathbb{E}_{x \sim \hat{P}}[\ell(\theta; x)], \tag{3.1}$$

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,

$$\min_{\theta \in \Theta} \left\{ \mathcal{R}(\theta) := \sup_{p \in \mathcal{Q}} \mathbb{E}_{x \sim p}[\ell(\theta; x)] \right\}, \tag{3.2}$$

where $\mathcal{Q}$ includes the potential test distributions that we hope the model performs well on, and p is a possible training distribution sampled from 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. (3.2) does not necessarily rely on the unknown test distribution p'. In the meantime, it is also the upper bound of the test risk [49], i.e., $\mathbb{E}_{p'}[\ell(x; \theta)] \leq \sup_{p \in \mathcal{Q}} \mathbb{E}_p[\ell(\theta; x)]$.

3.2 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 3.1.

3.2.1 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 [54]. 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,

$$\mathcal{Q} := \left\{ \sum_{g=1}^{m} q_g P_g : q \in \Delta_m \right\}, \tag{3.3}$$

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,

$$\hat{\theta}_{\text{DRO}} := \arg\min_{\theta \in \Theta} \left\{ \hat{\mathcal{R}}(\theta) := \max_{g \in \mathcal{G}} \mathbb{E}_{x \sim \hat{P}_g}[\ell(\theta; x)] \right\}, \tag{3.4}$$

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,

$$\min_{\theta \in \Theta} \sup_{q \in \Delta_m} \sum_{g=1}^{m} q_g \mathbb{E}_{(x) \sim P_g}[\ell(\theta; x)], \tag{3.5}$$

where q is a m-dimensional trainable vector with the same meaning as the one in Eq. (3.3). According to Eq. (3.5), 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 [49, 55, 58], which could be naturally adapted with the deep models trained in a supervised way for solving VRPs, such as GCN [14]. Compared with supervised learning, reinforcement learning (RL) is much preferred given its independence of optimal solution as the ground-truth label. For example, POMO [17] trained with reinforcement learning,

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 ℓ_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: Renormalize weights by $q^{(t)} \leftarrow q' / \sum_{g'} q'_{g'}$
13: **end for**

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 [87] 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,

$$\nabla \ell = \sup_{q \in \Delta_m} \sum_{g=1}^{m} q_g \mathbb{E}_{\substack{M(\theta;x) \\ x \sim P_g}} [(l(\pi) - b(x)) \nabla log M(\theta; x)], \tag{3.6}$$

where $b(x)$ denotes the baseline to reduce the variance of gradients. We maintain an 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 [58], 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.

3.2.2 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 [88, 89].

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. Specifically, 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.

3.3 Experiments

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.

3.3.1 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* [90, 91], and also normalize them to the $[0,1]^2$ 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. The number of groups is determined by the number of distributions in the training dataset.

We apply our approach to two different types of deep models for solving VRPs, i.e., GCN[1] [14] and POMO[2] [17] , 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 Concorde [92] and LKH3 [93] 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 done in [58]. Regarding the convolutional embedding layer by CNN, the input length for each node equals 2 (coordinate)

[1]https://github.com/chaitjo/graph-convnet-tsp
[2]https://github.com/yd-kwon/POMO

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 [94] 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 method are essentially different from the ones in their original works.

3.3.2 Comparison with Baselines

We proceed by conducting a series of experiments involving DROG and DROP, employing a diverse range of baselines for comparison. These baselines include: 1) Concorde, a specialized solver designed for exact solutions to the Traveling Salesman Problem (TSP); and 2) LKH3, a robust heuristic solver renowned for its efficacy across multiple variants of the Vehicle Routing Problem (VRP). 3) attention model (AM) [6], 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

Distribution		TSP50					TSP100				
	&Method	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
	AM	6.60	4.10%	7.11	6.28%	8m	9.01	5.01%	9.96	8.85%	31m
	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
	POMO	6.52	2.84%	7.05	5.38%	19s	8.84	3.03%	9.63	5.25%	1m
	DROP	**6.38**	**0.63%**	**6.71**	**0.30%**	18s	**8.73**	**1.75%**	**9.40**	**2.73%**	1m
Implosion	Solvers	6.40	0.00%	6.74	0.00%	15m	8.76	0.00%	9.50	0.00%	1h
	AM	6.64	3.75%	7.23	7.27%	8m	9.12	4.11%	10.21	7.47%	33m
	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
	POMO	6.57	2.66%	7.16	6.23%	22s	8.89	1.48%	9.80	3.16%	2m
	DROP	**6.43**	**0.47%**	**6.78**	**0.59%**	22s	**8.81**	**0.57%**	**9.58**	**0.84%**	2m
Expansion	Solvers	6.13	0.00%	6.19	0.00%	13m	8.19	0.00%	9.28	0.00%	1h
	AM	6.36	3.75%	6.55	5.82%	7m	8.46	3.30%	9.72	4.74%	28m
	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
	POMO	6.29	2.61%	6.61	6.79%	19s	8.40	2.56%	9.50	2.37%	1m
	DROP	6.20	1.14%	**6.23**	**0.65%**	19s	**8.34**	**1.83%**	9.35	0.75%	1m
Cluster	Solvers	5.88	0.00%	6.17	0.00%	10m	7.44	0.00%	7.61	0.00%	1h
	AM	6.12	4.08%	6.67	8.10%	6m	7.81	4.97%	7.97	4.73%	26m
	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
	POMO	6.07	3.23%	6.33	2.59%	18s	7.68	3.23%	7.80	2.50%	1m
	DROP	**5.90**	**0.34%**	**6.22**	**0.81%**	18s	**7.58**	**1.88%**	**7.73**	**1.58%**	1m
Grid	Solvers	6.02	0.00%	6.28	0.00%	11m	7.85	0.00%	7.99	0.00%	1h
	AM	6.24	3.65%	6.65	5.89%	7m	8.13	3.57%	8.51	6.51%	28m
	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
	POMO	6.20	2.99%	6.41	2.07%	18s	8.01	2.04%	8.32	4.13%	1m
	DROP	6.10	1.33%	6.33	0.80%	17s	**7.93**	**1.02%**	**8.09**	**1.25%**	1m

[1] The gap is computed based on Concorder.

[2] $\star$ average results over atypical instances; **Bold** best result from deep models.

TABLE 3.1: Results for Comparison with Baselines

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 also record the objective value and gap for the atypical instances to indicate the

Distribution	&Method	CVRP50 Obj.	Gap	Obj*	Gap*	Time	CVRP100 Obj.	Gap	Obj*	Gap*	Time
Explosion	Solvers	12.02	0.00%	12.30	0.00%	5h	17.18	0.00%	17.53	0.00%	11h
	AM	12.31	2.41%	12.83	4.31%	14m	17.79	3.55%	18.34	4.62%	52m
	GCN	-	-	-	-	-	-	-	-	-	-
	DROG	12.17	1.25%	12.50	1.63%	38m	**17.55**	**2.15%**	17.96	2.45%	2h
	POMO	12.26	2.00%	12.69	3.17%	32s	17.64	2.68%	18.14	3.48%	3m
	DROP	**12.15**	**1.08%**	**12.46**	**1.30%**	31s	**17.55**	**2.15%**	**17.95**	**2.40%**	3m
Implosion	Solvers	12.11	0.00%	12.45	0.00%	5h	17.25	0.00%	17.61	0.00%	10h
	AM	12.56	3.72%	12.93	3.86%	13m	17.70	2.61%	18.42	4.60%	52m
	GCN	-	-	-	-	-	-	-	-	-	-
	DROG	**12.26**	**1.24%**	**12.55**	**0.80%**	30m	17.58	1.91%	18.06	2.56%	3h
	POMO	12.43	2.64%	12.71	2.09%	30s	17.65	2.32%	18.25	3.63%	3m
	DROP	12.28	1.40%	12.60	1.20%	29s	**17.52**	**1.57%**	**18.02**	**2.33%**	3m
Expansion	Solvers	11.86	0.00%	12.14	0.00%	5h	16.98	0.00%	17.37	0.00%	9h
	AM	12.23	3.12%	12.69	4.53%	12m	17.66	4.00%	18.23	4.95%	44m
	GCN	-	-	-	-	-	-	-	-	-	-
	DROG	**11.95**	**0.76%**	**12.24**	**0.82%**	28m	17.32	2.00%	17.75	2.19%	2h
	POMO	12.08	1.85%	12.49	2.88%	26s	17.55	3.36%	18.04	3.86%	2m
	DROP	11.97	0.93%	12.28	1.15%	25s	**17.30**	**1.88%**	**17.73**	**2.07%**	2m
Cluster	Solvers	11.68	0.00%	11.83	0.00%	5h	16.69	0.00%	16.98	0.00%	8h
	AM	12.04	3.08%	12.47	5.41%	10m	17.30	3.65%	17.97	5.83%	40m
	GCN	-	-	-	-	-	-	-	-	-	-
	DROG	11.84	1.37%	12.10	2.28%	27m	17.09	2.40%	17.50	3.06%	2h
	POMO	11.88	1.71%	12.32	4.14%	23s	17.22	3.18%	17.73	4.42%	2m
	DROP	**11.79**	**0.94%**	**12.03**	**1.69%**	23s	**17.01**	**1.92%**	**17.42**	**2.59%**	2m
Grid	Solvers	11.72	0.00%	11.97	0.00%	5h	16.71	0.00%	17.05	0.00%	9h
	AM	12.14	3.58%	12.52	4.59%	10m	17.32	3.65%	17.90	4.99%	43m
	GCN	-	-	-	-	-	-	-	-	-	-
	DROG	11.83	0.94%	12.12	1.25%	27m	17.08	2.21%	17.51	2.70%	2h
	POMO	11.92	1.71%	12.35	3.17%	25s	17.21	2.99%	17.75	4.11%	2m
	DROP	**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 LKH3 for CVRP.

[2] $\star$ average results over atypical instances; **Bold** best result from deep models.

TABLE 3.2: Results for Comparison with Baselines

worst-case performance. All results are summarized in Table 3.1 and 3.2. We see that our models (DROG and DROP) substantially 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*).

	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 3.3: 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 3.4: Generalization Results on TSPLib and CVRPLib

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 3.1 and Table 3.2). 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 3.1 and Table 3.2.

3.3.3 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 3.3. 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 generalization 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.

3.3.4 Evaluation on Benchmark Dataset

We continue to evaluate DROG and DROP on the public benchmark dataset, i.e., TSPLib [95] and CVRPLib [96], whose instances may belong to various distributions that are totally distinct from ours. All learning-based models are trained

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 3.4. 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.

3.4 Chapter Summary

In this chapter, 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.

Chapter 4

Learning to Solve Routing Problems via Multi-View Graph Contrastive Learning

The Vehicle Routing Problem (VRP) is a vital combinatorial optimization challenge with applications in logistics. Often, similar VRP instances with distinct data occur repeatedly, such as daily package deliveries. Traditional heuristic methods treat these tasks independently, leading to limited efficiency and solution quality. Neural heuristics based on deep learning offer a solution, aiming to uncover underlying patterns to enhance performance. Early neural heuristics were supervised, relying on optimal solutions as labels. However, these methods are computationally demanding and struggle with generalization. Deep reinforcement learning-based heuristics use rewards instead, making them more adaptable to varying problem sizes. Although neural heuristics show promise, most are trained and evaluated on instances with uniform node distributions, limiting their applicability to diverse real-world instances.

Efforts have been made to address this generalization issue using group distributionally robust optimization (DRO) in the last chapter, along with another work named adaptive hardness-assisted curriculum learning (HAC) [48]. However, the former needs to label typical and atypical instances, and the latter is mainly extended to Gaussian distribution only. Hence, there is room for further improvement in methods in this line.

In this chapter, we propose Multi-View Graph Contrastive Learning (MVGCL) to improve neural heuristics' generalization capabilities for VRPs. MVGCL leverages contrastive learning to identify transferable local and global patterns across graphs. The pre-trained graph neural network (GNN) encoder captures node and graph embeddings, enhancing the neural heuristic's route construction.

4.1 Preliminaries

We consider the two classic VRP variants: travelling salesman problem (TSP) and capacitated vehicle routing problem (CVRP) in Euclidean space. Particularly, a problem instance with n nodes is represented as an undirected graph $G = (V, E)$ with complete connections. The cost c_{ij} for edge e_{ij} ($e_{ij} \in E$) denotes the distance between node v_i and v_j ($v_i, v_j \in V$). The vehicle in TSP needs to visit each node once and return to the starting one. In CVRP, a vehicle with capacity Q begins and ends its round trips at the depot node, while visiting each of other nodes once to satisfy customer demand d_i. The vehicle has to be fully replenished at the depot if the remaining capacity is not enough to serve any unvisited node. The optimal solution is defined as the route with the shortest length that complies with all the above constraints.

4.2 Methodology

We first present the graph representation of TSP and CVRP, followed by our multi-view graph contrastive learning (MVGCL) approach for solving the two problems.

4.2.1 Multi-view GCL for VRPs

When solving real-world VRP instances, an inevitable issue faced by neural heuristics is how to generalize the trained models to different distributions. In fact, most existing neural heuristics [6, 14, 17] for VRPs did not explicitly consider such matter and thus the underlying models trained on one distribution often yield inferior cross-distribution generalization.

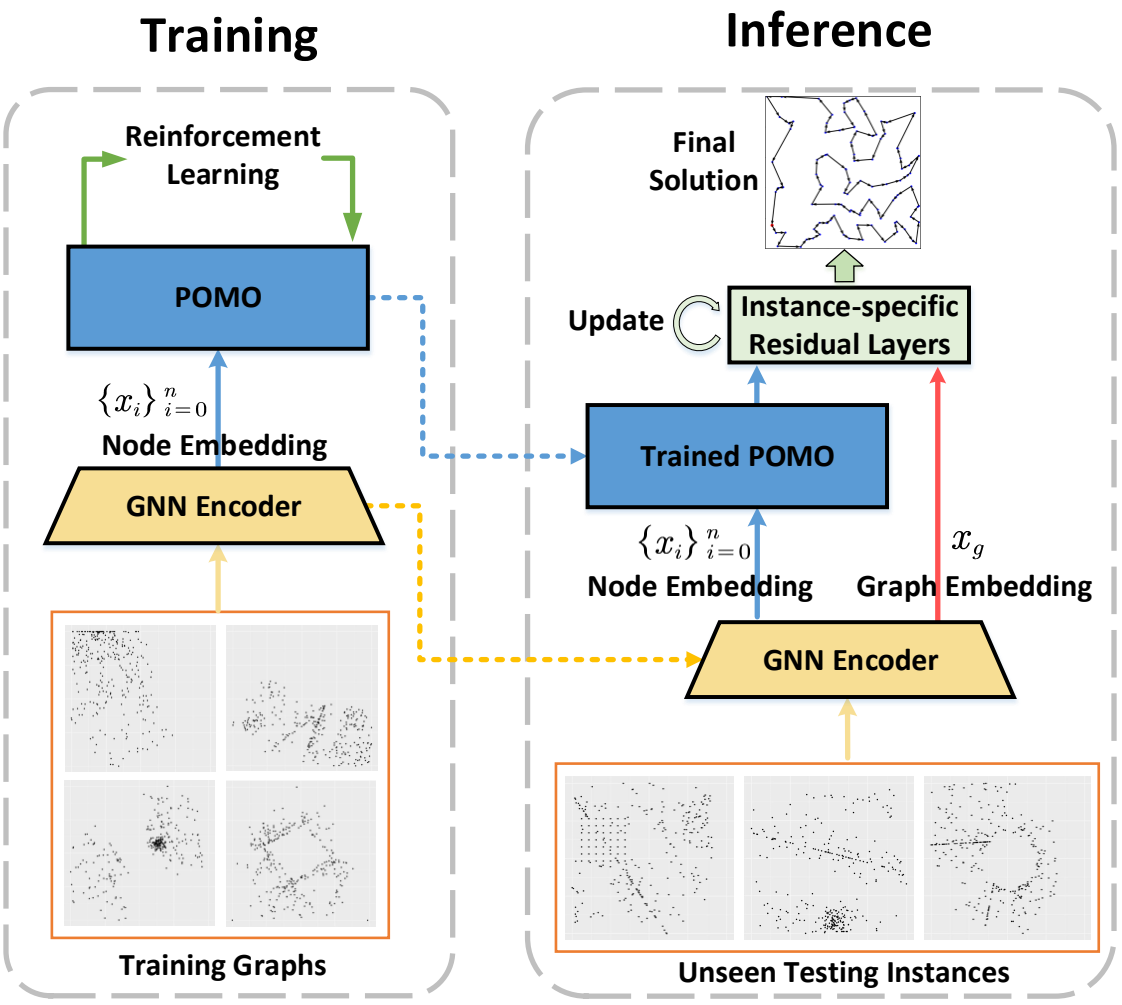

FIGURE 4.1: The illustration of our MVGCL, where the multi-view embeddings from the pre-trained GNN will be used to facilitate the subsequent training and inference.

To tackle this issue, we present a multi-view graph contrastive learning (MVGCL) approach for solving VRPs. Given graphs of VRP instances, our key idea is pre-training a GNN encoder to learn useful cross-distribution representations of nodes and graphs, which can be used to enhance the generalization of the neural heuristics. Unlike CV [60, 61] or NLP [63, 97], in which the common patterns could be similar image patches or words, the common patterns in VRP graphs are more obscure. From the geographic view, 1) a pattern could be clusters where customers concentrate on small areas; 2) it could also be hollows where only a few or even no customers exist. Rather than dominating the whole graph, these patterns may exist locally as small subgraphs across various distributions. Regarding the neural heuristics for routing problems [6, 17], explicitly (pre-)training over such

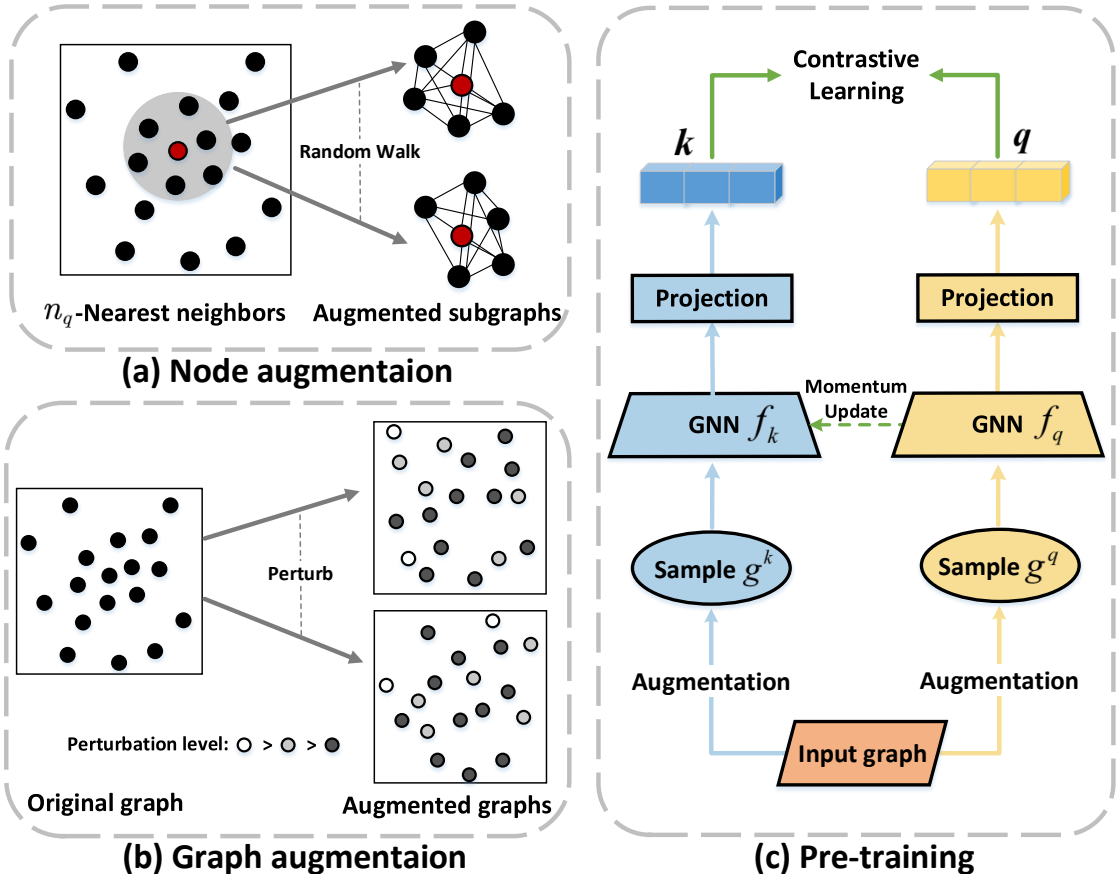

FIGURE 4.2: The framework of our node-level and graph-level contrastive learning, where the node augmentation in (a) and graph augmentation in (b) will share the same pre-training paradigm in (c) to attain the GNN f_q (f_k will be discarded).

patterns may help them learn more informative representation and potentially alleviate overfitting, thus fostering the generalization capability. To this end, in our MVGCL, we pre-train a GNN encoder on the corpus with various instance distributions for learning the representation of local structural patterns of nodes and global patterns of graphs. Then the learnt node embedding is employed to facilitate the neural heuristic (i.e. POMO [17]) trained with reinforcement learning, and the learnt graph embedding is employed to facilitate the active search [25] equipped to the neural heuristic during inference. The overall framework of our MVGCL is depicted in Figure 4.1.

4.2.2 Pre-training with GCL

In order to identify local patterns for nodes and global patterns for graphs, we exploit graph contrastive learning (GCL) to pre-train graph neural networks (GNN) in a self-supervised fashion. A contrastive learning framework for a specific problem usually comprises a properly defined contrastive objective and query/key samples. Regarding the former, we exploit the InfoNCE for subgraph as the objective function [59, 64, 98], given its fit to our problem. Regarding the latter, typically, there will be an encoded query $\boldsymbol{q}$ (a local or global pattern) and a dictionary with a set of K+1 encoded keys $\mathcal{K} = \{\boldsymbol{k}_i\}_{i=0}^{K+1}$, where both positive keys (similar patterns) and negative keys (dissimilar patterns) exist. The similarity between the query and keys is measured by the dot product between $\boldsymbol{q}^\top$ and $\boldsymbol{k}_i$. The positive key $\boldsymbol{k}_+$ should match the query $\boldsymbol{q}$, which shares a similar pattern as $\boldsymbol{q}$ and produces a high value for $\boldsymbol{q}^\top \boldsymbol{k}_+$. The similarity between $\boldsymbol{q}$ and negative keys should be low. Accordingly, the loss function of InfoNCE in our method is expressed as follows,

$$\mathcal{L}_q = -\log \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_+ / \tau\right)}{\sum_{i=0}^{K} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i / \tau\right)}, \tag{4.1}$$

where τ is a hyper-parameter of temperature that regulates the weights of penalties on negative samples [98]. Intuitively, the above (K+1)-element softmax loss function encourages the model to classify $\boldsymbol{q}$ as $\boldsymbol{k}_+$, which allows our GNN encoder to learn similar representations for nodes or graphs with similar patterns in the VRP instances.

4.2.2.1 Node-level representation learning

The performance of contrastive learning highly relates to how contrastive samples are defined for producing query $\boldsymbol{q}$ and keys $\mathcal{K}$. Pertaining to the general graphic tasks, it is natural to define samples as nodes with rich attributes or r-ego subgraphs produced based on the connectivity of nodes [64, 70]. However, these ideas cannot be directly applied to routing tasks. On the one hand, nodes in VRPs may not carry rich attributes, e.g., only coordinates for TSP (plus demands for CVRP). On the other hand, the connectivity is not informative for *complete* graphs of VRP. Given these points, we resort to geographic information to define the samples for contrastive learning.

Intuitively, the decision of visiting the next node in VRP is often subject to its geographical neighbourhoods [14, 45], where the probability for visiting a node in the same cluster should be higher than visiting others. It motivates us to strengthen the cross-distribution generalization by mining the local structural patterns. Instead of directly feeding coordinates of nodes into deep models as done in existing neural heuristics, we expect that a pre-trained GNN encoder could empower the neural heuristic with informative local structure representations of each node. In this sense, subgraphs around a node are deemed as natural choices to acquire positive and negative samples for contrastive learning. Therefore, we propose to discriminate subgraphs for different nodes as our pre-training task, where we feed them to the GNN encoder to produce representations for local structural patterns of the nodes.

We present the process of collecting positive and negative pairs of samples for each node. Specifically, as demonstrated in Figure 4.2 (a), we first extract a n_q-nearest-neighbours subgraph for a node and then apply the multi-hop random walk (MHRW) [99] on this subgraph to further generate MHRW subgraphs as the augmented samples, which are then fed into the GNN encoder. Since the MHRW subgraphs of the same node may contain similar customer sets and structures, we specify two samples augmented from the same n_q-nearest-neighbours subgraph as a positive pair, and those sampled from different nodes as negative ones. For our VRPs, we further specify the weight of walking from node v_i to v_j as $1/c_{ij}$ in MHRW, which is inversely proportional to the distance between them. This setting encourages to aggregate more structural information from the vicinity than non-vicinity, and such local patterns may potentially foster the generalization across distributions. In general, a VRP graph is a weighted undirected graph G with n nodes, and the cost c_{ij} ($c_{ij} > 0$) denotes the length of the edge $e_{i,j}$ between node v_i and node v_j, where we define $c_{ii} = 0$. The random walk on the graph G acts like that we roll dice at a node to decide which edge will be traversed next and lead to a new node. In order to sample a walk, we start from the anchor node, and then iteratively move from the current node to another node v_j within the n_q-neighbourhood-subgraph of the anchor node. In our implementation, the walk sampling depends on parameters that specify probabilities of walking to each node at the current step, i.e., the transit probability matrix $T = \{t_{ij}\}$. As stated in the main paper, we intuitively encourage more aggregation of the structural information from the vicinity than that from non-vicinity. Therefore, in our walk sampling,

we specify the transit probability t_{ij} from node v_i to node v_j as $1/c_{i,j}^{\alpha}$, where α is the hyperparameter controlling the importance of edge cost during the walk (i.e., $\alpha = 1$ in our setting). With the defined T, we iteratively visit the neighbourhood of the current node, until $\frac{3}{4}n_q$ nodes are collected for constructing a subgraph. Meanwhile, we hope the random walk concentrates more around the anchor node, and thus introduce a positive probability r (i.e., $r = 0.8$ in our setting) at each step for leading the walk back to the anchor node as did in [100].

During MHRW sampling, we collect the visited nodes to build the node set of the subgraph, and keep all edges between these nodes from the original graph. The generated subgraphs are then used as augmented samples to be processed by the GNN encoders for contrastive learning.

Meanwhile, we adopt the Momentum Contrast (MoCo) [59, 61] mechanism to pre-train the encoder with augmented samples in a contrastive way. In our approach, the MoCo includes an online GNN f_q and a smoothly-varying momentum GNN f_k as encoding networks with $\boldsymbol{q} = f_q\left(g^q\right)$ and $\boldsymbol{k} = f_k\left(g^k\right)$ (Figure 4.2 (c)), where g^q and g^k are the MHRW subgraphs sampled from the input graph. To keep consistent dictionary and stable training, the parameters θ_k of f_k are updated according to θ_q of f_q with a momentum coefficient $m \in [0, 1)$ as follows,

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q. \tag{4.2}$$

In each epoch, our approach samples a batch of g^q and g^k augmented from the same node as the positive pairs to produce q and k_+. The MoCo enqueues keys (i.e. the representations produced by f_k) from preceding mini-batches in each epoch to maintain a large dictionary without additional back-propagation costs. The InfoNCE loss in Eq. (4.1) is calculated with the q, k_+ and the key representations from the dictionary, and it only back-propagates to f_q. Then the node embeddings $\{x_i\}_{i=1}^n$ from this pre-trained GNN f_q will be used to facilitate generalizing the neural heuristic to various local structures of nodes for potentially reducing the route length on unseen VRP graphs.

4.2.2.2 Graph-level representation learning

From the multi-view perspective, the node embedding is more specific for local information, while the graph embedding may carry more useful global information. As verified in previous works on the selection of combinatorial optimization solvers [101, 102], the embedding of the global graph is important for the decision-making to a targeted instance. In our MVGCL, we propose a new augmentation mechanism for the graph embedding (as shown in Figure 4.2 (b)), which could limit the variance of distribution shift to a reasonable level and avoid alternating the distribution significantly as encountered in the general GCL augmentations [69, 70].

First, we normalize the coordinates of each node in the VRP graph to $[0, 1]^2$. To generate an augmented graph, for each node of the input VRP graph, we sample a perturbation level η from a categorical distribution, i.e., $\eta \in \{0.1, 0.2, 1\}$, where $\eta \sim \text{Categorical}(p_1, p_2, p_3)$, $p_1 + p_2 + p_3 = 1$, $p1 > p2 \gg p3$. Then, the perturbation of the coordinate $\mathbf{v}_i$ of node v_i, can be described as follows,

$$\mathbf{v}_i' = \mathbf{v}_i + \eta \cdot \Delta\mathbf{v}_i; \quad \Delta\mathbf{v}_i \sim \mathcal{U}\left[-\eta, \eta\right]^2, \tag{4.3}$$

where $\Delta\mathbf{v}_i$ is uniformly sampled in a square with length 2η. In this way, most (about $p_1 \times n$) nodes are relocated within its adjacent area $\Delta\mathbf{v}_i \sim \mathcal{U}\left[-0.1, 0.1\right]^2$; a small part (about $p_2 \times n$) of nodes are relocated within lager adjacent area $\Delta\mathbf{v}_i \sim \mathcal{U}\left[-0.2, 0.2\right]^2$; only a very few (about $p_3 \times n$) nodes are likely relocated to farther area $\Delta\mathbf{v}_i \sim \mathcal{U}\left[-1, 1\right]^2$. The idea behind this is that we keep the majority of nodes stay around the original position and thus the overall distribution is preserved after perturbation. To boost the generalization of this pre-trained model, we increase augmentation diversity by allowing the minority of nodes to shift farther.

Next, we randomly rotate the perturbed graphs. Then augmentations from the same input graph are deemed as positive pairs and those from different ones are negative pairs. Note that the graph representation learning share the same training paradigm as the node representation learning (i.e., Figure 4.2 (c)). During the training of MoCo [59], we feed those pairs into the encoders. Finally, GNN f_q learns to produce representations for similar distributions that are close to each other in the embedding space and also invariant to perturbations.

For solving VRPs, the inference based on active search iteratively updates the model parameters for each individual instance [16, 25], and the incorporation of global graph-aware information has the potential to further boost the performance. To this end, in the inference phase of the neural heuristic, the GNN encoder f_q passes the graph embedding x_g of the input graph, as the auxiliary information to favourably guide the active search.

4.2.3 Solving VRPs with pre-trained GNN

The goal of solving VRPs is to attain a valid trajectory for the given problem instance. Taking TSP as an example, the policy in reinforcement learning iteratively outputs an action a_t to select the next node to visit at step t, until all nodes have been included in the final trajectory $\pi = \{a_1, \cdots, a_n\}$. And the pre-trained GNN can be used in both policy optimization and active search in inference.

4.2.3.1 Policy optimization with node embedding

In the training phase, we use Policy Optimization with Multiple Optima (POMO) [17] as the neural heuristic to learn route construction step by step. POMO employs a self-attention encoder and an attention decoder to generate the solution (route) autoregressively. In our approach, instead of directly feeding coordinates into the attention network [17, 18, 48], we pass the node embeddings $\{x_i\}_{i=1}^n$ from the pre-trained GNN encoder f_q as the input to the self-attention encoder of POMO. In doing so, it is supposed to strengthen the generalization of POMO since more useful local structural information is injected. The attention decoder in POMO accepts the output of its self-attention encoder as the keys and values for its multi-head attention layers. Finally, a softmax layer computes the probabilities of visiting each node at the next step. To preserve desirable training performance, we follow the multiple greedy trajectories and instance augmentation with REINFORCE algorithm [87], as done in the original POMO.

4.2.3.2 Active search with graph embedding

In the inference phase, to better generalize the trained model on unseen instances, we exploit active search [16] to dynamically adjust the parameters for each target instance. The active search, known as an inference boosting mechanism, is originally proposed in [16], which actively updates the parameters of a model while it is used to infer the solution to an instance. Specifically, the inference starts with a trained model and iteratively optimizes its parameters with inferred solutions to an individual testing instance, while keeping track of the best one generated during the search (inference). This approach is verified to be competitive given a long runtime since it focuses on updating parameter for each individual instance. However, updating all parameters of the model as did in [16] is expensive and impractical. For example, it may cost several days to infer 10000 TSP100 instances. To tackle this issue, Efficient Active Search (EAS) is proposed in [25] to only adjust a subset of parameters during inference, while keeping all other parameters fixed.

We adopt a similar technique as EAS, which adds instance-specific residual layers before the output layer of the attention decoder (within POMO) and only updates the parameters of these layers. In our MVGCL, the residual layers accept both node embeddings from the last attention layer and the graph embedding x_g from the pre-trained encoder f_q, as formulated in the main paper. The instance-specific layers are updated with the aforementioned REINFORCE algorithm in POMO, but we also use the imitation loss J_{IL} to increase the log-likelihood of generating the incumbent solutions such that,

$$\nabla_{\theta'} J_{IL}(\theta) = \log p_{\nabla_{\theta'}} \left(\overline{\boldsymbol{\tau}} \mid s \right), \tag{4.4}$$

where θ' represents parameters of the instance-specific layers and $\overline{\tau}$ is the incumbent solution so far, i.e., the best solution till the current search iteration.

In our multi-view approach, the residual layers accept both node embeddings from preceding layers and the graph embedding x_g from the pre-trained encoder, as illustrated in Figure 4.1. Therefore, the representation of the entire graph could be exploited by the instance-specific layers to capture high-level information (e.g. the distribution of nodes) of the targeted VRP graph, so as to more effectively guide the active search to solve the corresponding instance. Particularly, the l-th

trainable residual layer is inserted into the attention decoder as

$$
\begin{aligned}
h_l &= \hat{h} + \left(\left(\mathrm{ReLu}\left(\hat{h}W_l^1 + b_l^1\right)W_l^2 + b_l^2\right)\right., \\
\hat{h} &= \begin{cases} [h_n, x_g] & , \ l = 1, \\ h_{l-1} & , \ l > 1, \end{cases}
\end{aligned} \tag{4.5}
$$

where W^1 and W^2 are the weight matrix; b^1 and b^2 are bias vectors; $\hat{h}$ of the first layer is the concatenation of the output of the last attention layer h_n and the graph embedding x_g from the pre-trained GNN.

4.3 Experiments

Our MVGCL is proposed to pre-train a GNN encoder to assist the neural heuristic in boosting the cross-distribution generalization performance. We evaluate our MVGCL on synthetic TSP and CVRP instances of various distributions, as well as those from the benchmark datasets, i.e., TSPLib and CVRPLib. We also conduct ablation studies to verify the respective key components of our MVGCL.

4.3.1 Experimental Settings

Baselines. For TSP, we compare with the exact solver Concorde [92] and representative neural heuristics including Attention Model (AM) [6], Policy Optimization with Multiple Optima (POMO) [17], Learning Collaborative Policies (LCP) [18], Distributionally Robust Optimization with POMO (DROP) [26], and Hardness-Adaptive Curriculum (HAC) [48], Efficient Active Search (EAS) [25]. Among them, DROP and HAC are recent works for tackling the generalization issue pertaining to routing problems. For CVRP, it is hard for existing solvers to attain optimal solutions in a reasonable time, we instead use the strong meta-heuristic HGS [103] as a conventional baseline, which reported superior performance to LKH3 [41]. Moreover, since HAC is originally designed for TSP only, we do not consider it for CVRP. We refer to the result of EAS in Table 4.4 for ablation study.

TABLE 4.1: Results of tour lengths and gaps to Concorde solver on various distributions (TSP)

Problem		TSP50							TSP100						
Distribution	**Metric**	Concorde	AM	POMO	LCP	HAC	DROP	MVGCL	Concorde	AM	POMO	LCP	HAC	DROP	MVGCL
Explosion	Len.	4.74	4.88	4.84	4.85	4.85	4.88	**4.80**	6.09	6.31	6.22	6.23	6.38	6.27	**6.17**
	Gap	0.00%	2.95%	2.11%	2.32%	2.32%	2.95%	**1.27%**	0.00%	3.61%	2.13%	2.30%	4.76%	2.96%	**1.31%**
Compression	Len.	5.22	5.37	5.33	5.32	5.35	5.34	**5.30**	6.89	7.16	7.06	7.07	7.18	7.12	**7.02**
	Gap	0.00%	2.87%	2.11%	1.92%	2.49%	2.30%	**1.53%**	0.00%	3.92%	2.47%	2.61%	4.21%	3.34%	**1.89%**
Cluster	Len.	5.37	5.56	5.50	5.54	5.53	5.52	**5.47**	7.26	7.58	7.45	7.48	7.63	7.46	**7.40**
	Gap	0.00%	3.54%	2.42%	3.17%	2.98%	2.79%	**1.86%**	0.00%	4.41%	2.62%	3.03%	5.10%	2.75%	**1.93%**
Expansion	Len.	4.44	4.58	4.55	4.56	4.58	4.60	**4.52**	5.57	5.80	5.72	5.78	5.88	5.74	**5.68**
	Gap	0.00%	3.15%	2.48%	2.70%	3.15%	3.60%	**1.80%**	0.00%	4.13%	2.69%	3.77%	5.57%	3.05%	**1.97%**
Rotation	Len.	4.54	4.69	4.64	4.68	4.66	4.64	**4.61**	6.02	6.28	6.17	6.20	6.34	6.23	**6.13**
	Gap	0.00%	3.30%	2.20%	3.08%	2.64%	2.20%	**1.54%**	0.00%	4.32%	2.49%	2.99%	5.32%	3.49%	**1.83%**
Avg. Inf. Time (s)		0.08	0.07	0.01	0.53	0.08	0.01	0.87	0.50	0.22	0.02	1.50	0.23	0.03	3.70

Implementation. We adopt the Graph Isomorphism Network (GIN) [104] as the encoders for f_q and f_k, while it is also free to try other GNN variants. We use the same hyperparameters as the original POMO [17], except that the batch size is reduced from 64 to 56 for CVRP100 due to the memory limit. We set $(p_1,\ p_2,\ p_3)$ to (0.7, 0.2, 0.1). During training, we apply *early stopping* when the gap reduction is not significant. We set the number of iterations in active search for each instance to 200.

Dataset. Instead of solely testing on uniform distribution as most existing neural heuristics did, we evaluate all methods on various distributions, i.e., Explosion, Compression, Cluster, Expansion and Rotation, respectively. These distributions are more visually and quantitatively diverse than the uniform one [90], thus intensifying the hardness for neural heuristics to generalize. To guarantee the essential diversity of local structural patterns and ensure that the (pre-)training instances are unseen in testing, we generate (pre-)training instances from mixed distributions by, 1) sampling an instance uniformly, 2) then randomly applying three non-repetitive mutation operators[1] from TSPGEN on this instance. We generate 6M such instances as the pre-training corpus of GCL and another 1.2M for training in our method. For other baselines without pre-training, we use all those 7.2M instances in their training phases. After training, we evaluate the models on 2000 instances from each of the above five testing distributions (i.e. 10000 instances in total). For HAC, we use its built-in data generator (i.e. hardness-adaptive generator) for the

[1]https://github.com/jakobbossek/tspgen/tree/master/R

best performance. We specify the last applied mutation as the class label for each instance, since the training of DROP needs this label.

4.3.2 Generalization on TSP

In Table 4.1, we display the average values of tour lengths (Len.), gaps (Gap.) to the optimal solutions (attained by Concorde solver) and the inference time, on the unseen instances from the five distributions for TSP50 and TSP100. Overall, the exact solver Concorde performs the best in terms of the tour length, since it is highly specialized for TSP. Among neural heuristic methods, our MVGCL achieves the smallest gap and significantly improves the generalization performance on the five testing distributions. For example, our MVGCL reduces the gap by 2.49% (1.83% vs 4.32%) on Rotation distribution of TSP100 compared to AM, and even for the strong neural baseline POMO, our MVGCL brings 0.82% (1.31% vs 2.13%) reduction of the gap on Explosion distribution of TSP100. While HAC fails to generalize well on TSP100 instances with relatively large gaps (4.21-5.32%), MVGCL consistently delivers smaller gaps (1.31-2.15%). Meanwhile, DROP exhibits unstable generalization performance, which might be caused by the training on instances of mixed-distributions without clear class division. Hence, our MVGCL outperforms the two state-of-the-art methods on cross-distribution generalization. Regarding the efficiency, Concorde, LCP and MVGCL iteratively improve solutions or adjust parameters, so induce longer runtime.

TABLE 4.2: Results of tour lengths and gaps to HGS solver on various distributions (CVRP).

Problem		**CVRP50**						**CVRP100**					
Distribution	**Metric**	HGS	AM	POMO	LCP	DROP	MVGCL	HGS	AM	POMO	LCP	DROP	MVGCL
Explosion	Len.	9.79	10.02	9.92	9.97	9.91	**9.81**	14.30	14.79	14.65	14.73	14.70	**14.33**
	Gap	0.00%	2.35%	1.33%	1.84%	1.23%	**0.20%**	0.00%	3.43%	2.45%	3.01%	2.80%	**0.21%**
Compression	Len.	10.14	10.39	10.28	10.35	10.32	**10.15**	14.82	15.36	15.21	15.30	15.32	**14.85**
	Gap	0.00%	2.47%	1.38%	2.07%	1.78%	**0.10%**	0.00%	3.64%	2.63%	3.24%	3.37%	**0.20%**
Cluster	Len.	10.35	10.60	10.49	10.57	10.49	**10.37**	15.44	15.99	15.83	15.89	15.90	**15.48**
	Gap	0.00%	2.42%	1.35%	2.13%	1.35%	**0.19%**	0.00%	3.56%	2.53%	2.91%	2.98%	**0.26%**
Expansion	Len.	9.40	9.64	9.53	9.60	9.61	**9.42**	13.70	14.18	14.02	14.14	14.19	**13.74**
	Gap	0.00%	2.55%	1.38%	2.13%	2.23%	**0.21%**	0.00%	3.50%	2.34%	3.21%	3.58%	**0.29%**
Rotation	Len.	9.43	9.66	9.56	9.64	9.58	**9.45**	13.97	14.46	14.30	14.42	14.39	**14.00**
	Gap	0.00%	2.44%	1.38%	2.23%	1.59%	**0.21%**	0.00%	3.51%	2.36%	3.22%	3.01%	**0.21%**
Avg. Inf. Time (s)		30	0.22	0.01	2.83	0.01	1.07	30	0.29	0.03	5.85	0.05	4.43

TABLE 4.3: Results on TSPLib and CVRPLib

Dataset	Metric	Opt.	AM	POMO	LCP	HAC	DROP	MVGCL
TSPLib	Len.	6.86	8.02	7.44	7.48	8.65	7.48	**7.05**
	Gap	0.00%	10.53%	5.16%	5.92%	16.75%	5.79%	**1.58**%
Avg. Time (s)		-	0.48	0.47	69.26	0.48	0.35	48.11
CVRPLib	Len.	16.97	17.82	17.71	17.83	-	17.84	**17.09**
	Gap	0.00%	6.05%	4.52%	5.23%	-	5.25%	**0.70%**
Avg. Time (s)		-	0.29	0.03	7.22	-	0.05	4.8

4.3.3 Generalization on CVRP

We display the results for CVRP50 and CVRP100 in Table 4.2. As aforementioned, it is much harder to find the optimal solution for CVRP, so we specify the heuristic solver HGS with runtime 30s as the baseline to compute the gaps. Despite the good performance of AM, POMO and LCP on uniform distribution (as reported in their original papers), we observe that they drastically deteriorate when generalizing to other distributions. For example, the strong neural baseline POMO reported gaps of 0.45% and 0.32% for CVRP50 and CVRP100 on uniform distribution in its original paper, whereas its gaps increase to about 2.5% on the five distributions. In fact, generalizing to different distributions on CVRP100 is challenging for neural baselines, which yield large gaps around 2.3%-3.7%. However, our MVGCL achieves significantly smaller gaps (by up to 10x) than those of neural baselines, which are comparable to HGS that runs much longer. The reason might be that our MVGCL is less sensitive to the varied distributions as it exploits the universal local patterns. To sum up, our method attains higher-quality solutions over most of the distributions.

4.3.4 Results on Benchmarks

We continue to evaluate our MVGCL on public benchmark datasets, i.e., TSPLib [95] and CVRPLib [105], to demonstrate that our method is also effective in addressing more realistic distributions. Regarding TSPLib, we solve the instances with 51-299 nodes. Regarding CVRPLib, we solve XML100 which contains 10000 CVRP100 instances with heterogeneous distributions. The coordinates in each of

TABLE 4.4: Ablation studies on MVGCL

Name	Component			TSP100	
	Node Embed.	Graph Embed.	EAS	Len.	Gap.
M1	✗	✗	✗	6.760	2.41%
M2	✓	✗	✗	6.732	1.98%
M3	✓	✗	✓	6.722	1.83%
M4	✗	✗	✓	6.729	1.94%
M5	✗	✓	✓	6.725	1.88%
M6′	✓	✓	✓	6.720	1.80%
M6	✓	✓	✓	**6.717**	**1.76%**

the above instances are normalized to [0, 1] for a fair comparison. Instead of training a new model for each testing instance set with identical distribution [25], we directly use the models trained on TSP100 and CVRP100 (with mixed distributions) to solve those instances.

The upper half of Table 4.3 shows the average results on TSPLib instances. It is revealed that our MVGCL can generalize well to real-world distributions and varied sizes, with a low gap (1.58%), which is significantly smaller than those of neural baselines (i.e. 5.16%-16.75%). The advantage of our MVGCL over HAC suggests that training with similar distributions (e.g. Gaussian mixture distributions) may limit the generalization performance, while pre-training with diverse patterns from various heterogeneous distributions could be more beneficial. The lower half of Table 4.3 shows the average results on CVRPLib instances, which reveal that our MVGCL can also generalize well to miscellaneous instances, which are completely unseen in training.

4.3.5 Ablation Studies

We further conduct ablation studies to verify the effectiveness of key components in our MVGCL, where we take TSP100 as an exemplary case. In Table 4.4, we ablate three components and report the average results over all 10000 instances of the five distributions. The comparison between M1 and M2 shows that the node embedding from our node-level GCL is helpful for generalization, which reduces the gap (2.41%) of the original POMO (M1) by 0.43%. The comparison between

M2 and M3 shows that the active search with additional neural layers (referred to as EAS) [25] can further reduce the gap by 0.15%. To verify the effectiveness of the distribution-preserved augmentation in our graph-level GCL, we use the pooling result of all node embeddings produced by the node-level GCL as the graph embedding x_g (M6′). We can only see a slight difference compared to M3, which implies that solely providing local information (M6′) cannot capture the overall distribution for more effective active search. In contrast, we can observe from M3 v.s. M6 and M4 v.s. M5 that guiding the active search by graph embedding produced by our graph-level GCL (M5 and M6) can significantly improve the generalization performance. Finally, our MVGCL with all components (M6) achieves the highest gap reduction (0.65%) compared to M1, which empirically verifies the importance of learning universal local patterns and the effectiveness of global graph embedding for active search.

4.4 Chapter Summary

In this chapter, we propose a multi-view graph contrastive learning approach to leverage node-level local patterns and graph-level global representation for neural heuristics equipped with active search to solve VRPs. Extensive experiments on synthetic instances and benchmark instances (TSPLib and CVRPLib) of various distributions show that our MVGCL significantly improves the cross-distribution generalization performance.

Although the proposed method effectively addresses routing problems under distribution shifts, its reliance on a single model constrains its performance. To mitigate this limitation, a logical step forward involves the integration of diverse models into an ensemble for solving VRPs across various distributions. The subsequent chapter outlines an ensemble-based deep reinforcement learning approach in this direction.

Chapter 5

Learning to Solve Routing Problems via Ensemble-based Deep Reinforcement Learning

Traditional methods solve VRPs independently, limiting efficiency and quality. Neural heuristics under Deep Reinforcement Learning (DRL) improve this, but they often lack cross-distribution adaptability. While the neural methods we proposed in the previous chapters demonstrated promising outcomes in generalization on VRPs, they still cannot limit their performance by only employing a single model. Notably, most of the recent deep learning-based approaches only consider single models [6, 17, 30, 39, 106–109], which could hinder their practical applications under complex situations. These single-model Neural heuristics based on deep learning enhance solutions of COPs but struggle with handling various distributions.

In this chapter, we highlight the limitations of simple ensemble approaches for VRPs with distribution shifts. We propose an Ensemble-based Deep Reinforcement Learning (EL-DRL) approach to address this challenge, where an ensemble of sub-policies collaboratively handles different distributions, bolstered by unique loss signals and inequality-based regularization. Our EL-DRL approach overcomes the above limitation by training an ensemble of sub-policies, each combination of them focusing on different distributions. We enhance the diversity of sub-models

by using varying loss signals and enforcing inequalities. EL-DRL adapts dynamically to various instances without manual specification. Using EL-DRL with the POMO sub-model, we can achieve superior cross-distribution generalization for VRPs across various instances.

5.1 Preliminaries

The traveling salesman problem (TSP) and the capacitated vehicle routing problem (CVRP) are the two most classic VRP variants. The TSP involves finding the optimal route to visit a set of N cities (or nodes) $\{v_1, v_2, \ldots, v_N\}$, with the objective to minimize the total distance traveled. The CVRP considers a fleet of (homogeneous) vehicles with limited capacity, where the vehicle must only visit a client once and also satisfy its demand. The objective is to minimize the total distance traveled by all vehicles while complying with the capacity constraint.

Our EL-DRL originates from the conjecture that high similarity in parameters of the neural network in the ensemble-based learning paradigm correlates to poor performance, as demonstrated in Section 5.3.4. Recent studies on ensembles with reinforcement learning [110–112] have primarily focused on utilizing deep Q-learning [113] to play games with a limited number of actions determined by the game controller. However, estimating Q-values for VRPs with hundreds of potential actions (i.e., the next node to visit) is challenging [44, 114], and the performance of these methods is often inferior to that of policy gradient [115]. We bridge this gap by designing the ensemble-based DRL (EL-DRL) to solve VRPs with policy gradient. While we build EL-DRL on top of the policy optimization with multiple optima (POMO) [17], our method is generic and can also be applied to other neural heuristics.

For clarity, we begin by explaining the process of using a single policy to solve a TSP instance through reinforcement learning. In specific, the policy gradient aims to learn a policy π_θ represented by a neural network and parameterized by θ, which constructs a solution for the input instance s. The (partial) solution τ_t is defined by the sequence of selected actions $a_1, a_2, \ldots, a_N$. Typically, the neural network employs an encoder to embed the input features (e.g. coordinates of nodes) and a

decoder to sample the next action a [6, 18]. To solve TSP, the decoder autoregressively outputs the probability distribution $\pi_\theta(\cdot \mid s, \tau_t)$ for selecting a valid node (action) to visit at each step $t \in \{1, \ldots, N\}$ such that $\pi_\theta(\tau) = \prod_t \pi_\theta(a_{t+1} \mid s, \tau_t)$. Once a complete solution τ has been constructed, the negative length of the trajectory can be computed and used as the reward $R(\tau)$.

5.2 Methodology

In this section, we present how our ensemble learning paradigm trains a set of diverse DRL-based sub-policies, i.e., EL-DRL (see Figure 5.1), to solve VRP instances of various distributions effectively.

5.2.1 Ensemble-based Policy Gradient

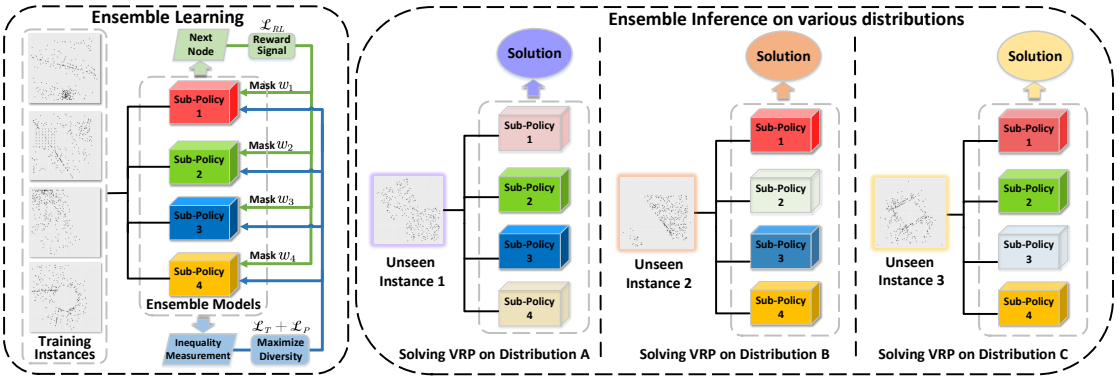

FIGURE 5.1: Illustration of Our EL-DRL. It trains sub-policies using respective masked reward (reinforcement loss) signals and inequality regularization to maximize the diversity. During inference, sub-policies synergize by leveraging their strengths on various distributions. Different levels of transparency indicate varied contributions of each sub-policy in deriving the solutions.

Our architecture employs a shared attention encoder from AM [6], followed by M respective attention decoders, as illustrated in Figure 5.1. Each sub-policy, denoted by π_θ^m with parameters θ_m, corresponds to a unique decoder plus an encoder that shares parameters with other sub-policies. In this way, the overall representation of an instance is only calculated once through the (computationally intensive) encoder. The subsequent lightweight decoders reuse this representation and collaboratively select nodes autoregressively, reducing the computational overhead for the ensemble training.

In our EL-DRL, the action a_t at each step is determined by gathering output from all sub-policies, i.e, $\pi_\theta(a_{t+1} \mid s, \tau_t) = \frac{1}{M}\sum_1^M \pi_\theta^m(a_{t+1} \mid s, \tau_t)$. This is inspired by the real-world practice where open-source communities combine the strengths of different individuals to achieve various common goals [116]. Similarly, our ensemble policies leverage the strengths of individual sub-models, e.g., some of them may perform well on Uniform distribution, while others excel on Gaussian distribution. Consequently, EL-DRL is not dominated by any individual sub-policy. Instead, it averages the probabilities produced by all sub-policies to determine an action at each step.

We train the sub-policy with the REINFORCE algorithm [87] based on the POMO model[17], which samples a set of solution trajectories $\left\{\tau^1, \tau^2, \ldots, \tau^N\right\}$ that start from each of all nodes ($i \in \{1, 2, \ldots, N\}$), and gather each return $R\left(\tau^i\right)$ (i.e., the negative of route length). The goal is to maximize the reinforcement learning objective $\mathcal{L} = \mathbb{E}_{\tau^1 \sim \pi_\theta} R\left(\tau^1\right) \mathbb{E}_{\tau^2 \sim \pi_\theta} R\left(\tau^2\right) \ldots \mathbb{E}_{\tau^N \sim \pi_\theta} R\left(\tau^N\right)$ by updating θ to increase the likelihood of sampling good trajectories in the future. We extend this objective to multiple sub-policies, which maximizes the expected return $\mathcal{L}_{RL}$ with gradients computed as below,

$$\nabla \mathcal{L}(\theta \mid s) \approx \frac{1}{NM} \sum_{i=1}^{N} \sum_{m=1}^{M} \left(R\left(\tau_m^i\right) - b(s)\right) \nabla \log \pi_\theta^m\left(\tau_m^i \mid s\right), \tag{5.1}$$

where $b(s)$ is a baseline to reduce the gradient variance, $\pi_\theta^m\left(\tau_m^i \mid s\right)$ means the probability produced by the m-th sub-policy for the solution τ_m^i given the instance s. Our sub-policies synergize to generate a single trajectory for each starting node i, thus we have $\tau_m^i = \tau^i$ and $R(\tau_m^i) = R(\tau^i)$ for all m.

While it is feasible to train all the sub-policies together with Eq. (5.1), directly applying it to them may lead to the identical convergence of parameters, because all sub-policies receive the same reward which is used to compute the reinforcement loss signal. To alleviate this issue, we harness the Bootstrap with random initialization [117] for training the ensemble of sub-policies, which fosters diversity across the sub-policies via two mechanisms. Firstly, we initialize all decoders with distinct random parameters, inducing an initial diversity in the ensemble. Secondly, we leverage distinct reinforcement loss signals to train each sub-policy. In particular, for a sub-policy π_θ^m, we generate binary masks w_m^i from a Bernoulli distribution with parameter $\beta \in (0, 1]$ and then apply them to the rewards. Integrating those

designs into Eq. (5.1), we arrive at:

$$\nabla \mathcal{L}_{RL}(\theta \mid s) \approx \frac{1}{NM} \sum_{i=1}^{N} \sum_{m=1}^{M} w_m^i \left(R\left(\tau^i\right) - b(s) \right) \nabla \log \pi_\theta^m \left(\tau^i \mid s\right). \tag{5.2}$$

When updating the decoder parameters of sub-policies, we also apply the bootstrap mask to each objective function, similar to the above one in Eq. (5.2). Additionally, we utilize a shared baseline $b(s)$ averaged from all sampled trajectories:

$$b(s) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{m=1}^{M} R\left(\tau_m^i\right) = \frac{1}{N} \sum_{i=1}^{N} R\left(\tau^i\right). \tag{5.3}$$

The probabilities of producing the actions of τ^i are reinforced through other N-1 unique trajectories. This shared baseline contributes to reducing gradient variance, and also potentially helps restrain the identical convergence with respect to the sub-policies [17].

5.2.2 Diversity Enhancement via Regularization

To further promote the diversity of the ensemble of sub-policies, we exploit the regularization to encourage explicit diversity across the parameters $\theta^1, \ldots, \theta^M$ of sub-policies during training. In order to prevent sub-policies from converging to the same parameters or simply duplicating each other, we need to push them away from each other in the parameter space.

This motivates us to measure and pursue the inequality between sub-policies. Therefore, we utilize a statistical measurement of inequality within a population (ensembles) in economics, i.e., the Theil index [118]. From the perspective of information theory, it is also a measurement of redundancy, which indicates the difference between the maximum possible entropy of the data and the observed entropy [86]. With the Theil index, we could measure whether our ensembles have learned their respective knowledge instead of imitating each other. Concretely, the first regularization term based on the Theil index in our loss function is expressed as $\mathcal{L}_T = \frac{1}{M} \sum_{m=1}^{M} \frac{x_m}{\mu} \ln \frac{x_m}{\mu}$, where x_m is the ℓ^2-norm of θ_m and μ is the mean of all ℓ^2-norms.

Furthermore, since we rely on an ensemble of sub-policies to make the decision, we could also explicitly exert a penalty that discourages all the individual sub-policies from converging toward a common value. This inspires us to utilize the second regularization term from the principles of consensus optimization [119], where multiple models are independently optimized with their own task-specific parameters. Formally, this regularization term is defined as $\mathcal{L}_P = \sum_{m=1}^{M} \|\bar{\theta} - \theta^m\|^2$, where $\bar{\theta}$ is the mean of the parameters for all sub-policies.

Notice that we intend to minimize the objective function for training our EL-DRL, thereby we integrate the regularization terms with negative signs to maximize the ensemble diversity (inequality). The whole loss function, inclusive of regularization terms with their coefficients (α_1 and α_2) for our EL-DRL is expressed as:

$$\nabla\mathcal{L} \approx \mathcal{L}_{RL}(\theta \mid s) - \alpha_1\mathcal{L}_T - \alpha_2\mathcal{L}_P. \tag{5.4}$$

At last, the parameters of policies are updated using gradient ascent based on Eq. (5.4). In the inference phase, EL-DRL collaboratively and greedily produces multiple trajectories by augmenting each input instance with rotations [17] and starting the trajectory from each of all nodes. The final solution is specified as the best one among all the sampled trajectories (solutions).

5.3 Experiments

We propose EL-DRL to train multiple diverse policies for enhancing cross-distribution generalization performance. To evaluate the robustness of our method in handling distribution shifts, EL-DRL and other baselines are tested on synthetic TSP and CVRP instances from various distributions, together with benchmark instances from TSPLib [95] and CVRPLib [105]. Additionally, we perform ablation experiments to demonstrate the importance of diversity and the key designs of our EL-DRL.

Baselines. For TSP experiments, we evaluate against the Concorde exact solver [92] and representative end-to-end neural approaches: Attention Model (AM) [6], Policy Optimization with Multiple Optima (POMO) [17], Multi-Decoder Attention Model (MDAM)[10], Distributionally Robust Optimization with POMO

(DROP) [26], Hardness-Adaptive Curriculum (HAC) [48], and Adaptive Multi-Distribution Knowledge Distillation (AMDKD) [20]. Of these methods, DROP, HAC and AMDKD are recent approaches tackling the cross-distribution generalization issue for solving VRPs. As HAC was initially intended only for TSP, we did not take it into account for our CVRP experiments.

Implementation. EL-DRL employs an architecture similar to AM [6] but with one encoder and four decoders as the backbone. Preliminary experiments showed that the use of four decoders allows for a good balance between computational overhead and performance. Each sub-policy was trained using the same hyperparameters as the original POMO [17]. We use Adam optimizer with a learning rate of 1e-4. The values of α_1 and α_2 are set to 0.5. All the experiments of neural baselines are conducted in parallel on NVIDIA GTX 2080Ti GPUs for both training and inference. The training time for EL-DRL varied with the problem size. Taking CVRP100 as an example, one epoch takes about 5 minutes for EL-DRL. We applied early stopping during training when the reduction in the gap was not significant.

The optimal solution for TSP is calculated using the Concorde solver.[1] For CVRP, as existing solvers are difficult to obtain optimal solutions within an acceptable time frame, we employ the strong heuristic HGS[2] [103] as the traditional baseline instead, which has reported better performance [41] than LKH3 [120]. Regarding neural baselines, we reuse their open-sourced code from GitHub. For AM-based baselines, we sample 1,280 solutions following [6]; for MDAM we set beam search size to 50; for POMO-based baseline, we adopt the greedy rollout with $\times 8$ augments following [17].

Dataset. To ensure the hardness and variety of training distributions and to guarantee that the training instances are not seen during testing, we generate 7.2 million training instances from mixed distributions. Data generation is done by first sampling a uniform instance and then non-repetitively applying three randomly mutation operators[3] of TSPGEN to the same instance. So a single training instance is of mixed distributions. During the inference, unlike most existing neural

[1] https://www.math.uwaterloo.ca/tsp/concorde.html
[2] https://github.com/chkwon/PyHygese
[3] https://github.com/jakobbossek/tspgen/tree/master/R

heuristics that are only tested on a uniform distribution, we evaluated all methods on various distributions, including Explosion, Compression, Cluster, Expansion, and Rotation, to assess the generalization performance against distribution shift. These distributions are more diverse in terms of both visual and quantitative measures compared to the uniform distribution [90], making it fairly challenging for neural heuristics to generalize. We test baselines on 10,000 instances in total, with each of the five testing distributions containing 2,000 instances. HAC is trained on instances created by the built-in data generator as its design relies on this process. The training of DROP requires class labels of the input instance and we specify the class of the last applied mutation operation for it. AMDKD requires manually assigning a single distribution for training each teacher model, which is less applicable to our setting where each single training instance is of mixed distributions. Therefore, we follow its original setting for the best performance, and directly utilize the pre-trained AMDKD to solve instances from TSPLib and CVRPLib (Table 5.5).

5.3.1 Cross-distribution Generalization on TSP

Table 5.1: Results of cross-distribution generalization on TSP50

Distribution	Metric	Concorde	AM	POMO	MDAM	HAC	DROP	EL-DRL
Explosion	Len.	4.74	4.88	4.84	4.87	4.85	4.88	**4.83**
	Gap	0.00%	2.95%	2.11%	2.74%	2.32%	2.95%	**1.90%**
Compression	Len.	5.22	5.37	5.33	5.36	5.35	5.34	**5.32**
	Gap	0.00%	2.87%	2.11%	2.68%	2.49%	2.30%	**1.92%**
Cluster	Len.	5.37	5.56	5.50	5.54	5.53	5.52	**5.48**
	Gap	0.00%	3.54%	2.42%	3.17%	2.98%	2.79%	**2.05%**
Expansion	Len.	4.44	4.58	4.55	4.57	4.58	4.60	**4.53**
	Gap	0.00%	3.15%	2.48%	2.93%	3.15%	3.60%	**2.03%**
Rotation	Len.	4.54	4.69	4.64	4.67	4.66	4.64	**4.63**
	Gap	0.00%	3.30%	2.20%	2.86%	2.64%	2.20%	**1.98%**
Avg. Inf. Time (s)		0.08	0.07	0.01	0.04	0.08	0.01	0.02

We evaluated the cross-distribution generalization of our EL-DRL on TSP50 and TSP100. Tables 5.1 and 5.2 gathered the averaged tour length and gaps of all baselines compared to the exact solver Concorde on unseen instances from five different distributions. Although all baselines were trained on mixed distributions, they did not generalize well to other specific distributions (especially for AM), resulting in

TABLE 5.2: Results of cross-distribution generalization on TSP100

Distribution	**Metric**	Concorde	AM	POMO	MDAM	HAC	DROP	EL-DRL
Explosion	Len.	6.09	6.31	6.22	6.29	6.38	6.27	**6.21**
	Gap	0.00%	3.61%	2.13%	3.28%	4.76%	2.96%	**1.97%**
Compression	Len.	6.89	7.16	7.06	7.12	7.18	7.12	**7.03**
	Gap	0.00%	3.92%	2.47%	3.34%	4.21%	3.34%	**2.03%**
Cluster	Len.	7.26	7.58	7.45	7.53	7.63	7.46	**7.41**
	Gap	0.00%	4.41%	2.62%	3.72%	5.10%	2.75%	**2.07%**
Expansion	Len.	5.57	5.80	5.72	5.78	5.88	5.74	**5.69**
	Gap	0.00%	4.13%	2.69%	3.77%	5.57%	3.05%	**2.15%**
Rotation	Len.	6.02	6.28	6.17	6.24	6.34	6.23	**6.14**
	Gap	0.00%	4.32%	2.49%	3.65%	5.32%	3.49%	**1.99%**
Avg. Inf. Time (s)		0.50	0.22	0.02	0.16	0.23	0.03	0.04

drastically deteriorated performance (about 2.5% to 4% gaps) compared to the i.i.d. testing results they reported in the original paper. For example, POMO reports only 0.03% and 0.14% gaps compared to LKH3 [120] on TSP50 and TSP100, respectively, when trained and tested on a uniform distribution in [17]. However, its gaps increase up to 2.69% in our cross-distribution experiments. This phenomenon implies that these neural baselines are relatively incapable of learning from the underlying common patterns on complex training distributions. In contrast, our EL-DRL significantly improved generalization performance on the five respective testing distributions, achieving the smallest gap among neural heuristic baselines. For example, our EL-DRL reduced the gaps by 2.34% (2.07% vs. 4.32%) compared to AM and achieved a 0.55% (2.07% vs 2.62%) gap reduction on the TSP100 with Cluster distribution compared to the best neural baseline POMO. Notice that while multiple decoders in MDAM slightly improved its performance compared to AM, it failed to generalize to various distributions with gaps exceeding 3.2% on TSP100. This indicates that simply composing multiple models (even with KL divergence) does not necessarily result in good generalization performance. Meanwhile, EL-DRL also consistently beats the recent generalization-specialized DROP and HAC in terms of both gaps and time efficiency. Especially, DROP exhibits unstable generalization performance and HAC does not generalize well on TSP100 (4.21-5.32% gaps), possibly due to special data processing requirements for training, which hinder their application to our scenario. In short, our EL-DRL outperforms these two state-of-the-art approaches in generalizing to cross-distribution.

5.3.2 Cross-distribution Generalization on CVRP

TABLE 5.3: Results of cross-distribution generalization on CVRP50

Distribution	Metric	HGS	AM	POMO	MDAM	DROP	EL-DRL
Explosion	Len.	9.79	10.02	9.92	9.98	9.91	**9.88**
	Gap	0.00%	2.35%	1.33%	1.94%	1.23%	**0.82%**
Compression	Len.	10.14	10.39	10.28	10.35	10.32	**10.24**
	Gap	0.00%	2.47%	1.38%	2.07%	1.78%	**0.69%**
Cluster	Len.	10.35	10.60	10.49	10.56	10.49	**10.45**
	Gap	0.00%	2.42%	1.35%	2.03%	1.35%	**0.87%**
Expansion	Len.	9.40	9.64	9.53	9.61	9.61	**9.50**
	Gap	0.00%	2.55%	1.38%	2.23%	2.23%	**0.85%**
Rotation	Len.	9.43	9.66	9.56	9.62	9.58	**9.53**
	Gap	0.00%	2.44%	1.38%	2.01%	1.59%	**0.74%**
Avg. Inf. Time (s)		30	0.22	0.01	0.13	0.01	0.02

TABLE 5.4: Results of cross-distribution generalization on CVRP100

Distribution	Metric	HGS	AM	POMO	MDAM	DROP	EL-DRL
Explosion	Len.	14.30	14.79	14.65	14.76	14.70	**14.59**
	Gap	0.00%	3.43%	2.45%	3.22%	2.80%	**2.03%**
Compression	Len.	14.82	15.36	15.21	15.28	15.32	**15.13**
	Gap	0.00%	3.64%	2.63%	3.10%	3.37%	**2.09%**
Cluster	Len.	15.44	15.99	15.83	15.91	15.90	**15.76**
	Gap	0.00%	3.56%	2.53%	3.04%	2.98%	**2.07%**
Expansion	Len.	13.70	14.18	14.02	14.17	14.19	**13.97**
	Gap	0.00%	3.50%	2.34%	3.43%	3.58%	**1.97%**
Rotation	Len.	13.97	14.46	14.30	14.40	14.39	**14.25**
	Gap	0.00%	3.51%	2.36%	3.08%	3.01%	**2.00%**
Avg. Inf. Time (s)		30	0.29	0.03	0.17	0.05	0.08

Pertaining to CVRP, we present the averaged tour lengths and gaps to the best solutions (obtained from the HGS with a 30s runtime), along with the average time to inference on unseen instances of CVRP50 and CVRP100 from five distributions in Tables 5.3 and 5.4, respectively. Overall, the HGS solver performs the best regarding tour length, as this heuristic is highly specialized for CVRP. Notably, we observe greater advantages of our method against neural baselines on this harder problem than on TSPs. Our EL-DRL achieves superior results on unseen instances of various distributions, with significantly small gaps (0.69-0.87%) on CVRP50, and especially outperforms POMO by around 0.7% on Compression distribution. Table 5.4 shows that for neural baselines, generalizing to various distributions

on CVRP100 is also challenging, with considerable gaps ranging from 2.36% to 3.64%. In contrast, our method consistently delivers high-quality solutions with shorter lengths and is less sensitive to distribution shifts, which attains gaps of around 2.0% for cross-distribution generalization on CVRP100. This superiority verifies the effectiveness of our idea of leveraging the strength of handling different distributions through cooperation between sub-policies. Regarding efficiency, all neural baselines can solve an instance in less than one second. Owing to our design of lightweight ensemble architecture and parallel inference, EL-DRL does not incur much inference time than the counterparts using a single model while yielding better performance and even exhibiting faster inference than AM and MDAM.

5.3.3 Real World benchmarks

TABLE 5.5: Results on Real-World Benchmarks

Dataset	**Metric**	Opt.	AM	POMO	MDAM	HAC	DROP	AMDKD	EL-DRL
TSPLib	Len.	6.86	8.02	7.44	7.73	8.65	7.48	7.45	**7.39**
	Gap	0.00%	10.53%	5.16%	7.82%	16.75%	5.79%	5.30%	**4.79%**
Avg. Time (s)		-	0.48	0.47	0.36	0.48	0.35	0.53	0.95
CVRPLib	Len.	16.97	17.82	17.71	17.76	-	17.84	17.67	**17.60**
	Gap	0.00%	6.05%	4.52%	4.90%	-	5.25%	4.32%	**3.89%**
Avg. Time (s)		-	0.29	0.03	0.15	-	0.05	0.03	0.08

We continue to evaluate our EL-DRL by taking public benchmark datasets TSPLib [95] and CVRPLib [105] as the testbed, to prove our approach is also efficient in solving more realistic and varied distributions. TSPLib is a library of TSP instances from various sources and of various types, making it desirable yet challenging for assessing generalization. For TSPLib, results are collected from 25 instances which problem sizes ranging from 50 to 300. For CVRPLib, the dataset is XML100, which contains 10,000 CVRP100 instances from heterogeneous distributions. We normalize the coordinates in each instance to a range of [0, 1] for all neural methods, which also include the neural baseline AMDKD [20].

The upper half of Table 5.5 illustrates the averaged results on TSPLib instances. These results demonstrate that EL-DRL generalizes well to realistic distributions and varied sizes, in a small gap (4.79%), and it is much lower than the gaps

of neural heuristics (i.e., around 5%-17%). In particular, the advantage of our EL-DRL over MDAM implies learning similar ensembles may hold back the generalization performance. Compared to the poor results of HAC, which learns only from Gaussian distribution, we can conclude that training the models on various heterogeneous distributions could be more beneficial. In comparison to the most recent generalization-specialized AMDKD, our EL-DRL brings a reduction in the gap by 0.51%. The lower half of Table 5.5 shows the averaged lengths and gaps on (harder) CVRPLib instances. Our EL-DRL again outperforms the top two baselines, AMDKD and POMO, by margins of 0.43% and 0.63%, respectively. The results on the realistic benchmarks underscore the remarkable capability of our EL-DRL in solving miscellaneous instances that were not present during the training phase, highlighting its robustness in coping with distribution shifts.

5.3.4 Ablation Studies

Table 5.6: Ablation Studies on CVRP100

Method	POMO	Multi-POMO	EL-DRL	w/o BRI	w/o $\mathcal{L}_T$	w/o $\mathcal{L}_P$
Avg. Length	14.801	14.784	**14.738**	14.768	14.752	14.756
Avg. Gap	2.423%	2.311%	**1.993%**	2.201%	2.090%	2.118%

We also carry out ablation studies to demonstrate the effectiveness of key designs in our EL-DRL, by taking CVRP100 as an exemplary case and presenting the results in Table 5.6. To investigate the disadvantage of high similarity between ensemble policies, we deploy a single POMO and an ensemble of four POMOs (Multi-POMO), respectively, to compare with our EL-DRL. Additionally, we ablate Bootstrap with Random Initialization (BRI), the Theil index regularization ($\mathcal{L}_T$) and parameter dissimilarity regularization ($\mathcal{L}_P$), then report the averaged results for the 10,000 instances of five distributions on CVRP100. First, the single POMO model relies on its own learning ability, increasing the susceptibility to over-fitting and leading to poor generalization performance. However, simply stacking POMO to multiple copies (Multi-POMO) cannot overcome this limitation and only brings slight improvement compared to the single POMO, which is primarily attributed to the larger amount of model parameters and in accordance with the finding in [121]. By contrast, our EL-DRL attains a 0.43% reduction of the overall gaps compared to the single POMO by collaboratively solving instances

on various distributions. Excluding the Bootstrap with random initialization, sub-policies cannot receive distinct loss signals and converge to similar ones, and thus the gap of the variant without BRI increases significantly to 2.201% compared to that of EL-DRL (1.993%). Moreover, removing two regularization terms for maximizing diversity results in redundant parameters for sub-policies and hinders their expressive ability, as evidenced by the drop in generalization performance shown in the last two columns of Table 5.6. In summary, the designs of our EL-DRL are largely orthogonal and can be effectively integrated to maintain ensembles of diverse sub-models, which are capable of capturing different aspects of various distributions.

5.4 Chapter Summary

In this chapter, we propose an ensemble-based DRL approach for solving vehicle routing problems, which fosters the capability of existing neural heuristics in generalizing against distribution shifts. By enforcing distinct reinforcement loss signals on each sub-policy and integrating regularization to ensure diversity across sub-policies, our EL-DRL learns to leverage the strengths of individuals in solving different instances with various distributions. In our experiments, we demonstrate that our proposed method delivers high-quality solutions for both synthetic instances of various distributions and heterogeneous instances from real-world benchmarks, confirming the potential of the ensemble idea in contributing to more generalizable and robust neural heuristics.

Although it is not the focus of this chapter, our method might be less effective in handling large instances. A major reason is that the scalability of our EL-DRL highly depends on the sub-model, where POMO (the used sub-model) performs inferior on large ones. Moreover, since our EL-DRL follows the ensemble paradigm, the training needs more computational resources than the single one. In the future, we would like to, 1) foster the scalability of our methods by exploring decomposition schemes such as the divide-and-conquer [122], so as to allow it to solve large instances more efficiently; 2) enhance the computational efficiency of our method by introducing sparsity in ensemble learning [123], so as to allow it to train more

sub-models while reducing the training overhead; and 3) enable the interpretability [124] and further strengthen the collaboration among the sub-models of our method.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis introduces an array of neural-based combinatorial optimization methods designed to autonomously learn heuristics for solving an extensive spectrum of distributions in Vehicle Routing Problems (VRPs).

In the first method, we harness the utility of group Distributionally Robust Optimization (DRO) to enhance the capacity of deep models in addressing Vehicle Routing Problems (VRPs) with heightened efficacy across diverse distributions. Additionally, we employ a Convolutional Neural Network (CNN) to exact foundational distribution-aware representations of VRP instances. This methodology is readily applicable with either supervised or reinforcement learning and evaluated with two well-known deep models, i.e., GCN and POMO. Empirical findings demonstrate the substantial augmentation of cross-distribution generalization performance achieved by our approach, surpassing learning-based baselines across synthesized and benchmark datasets. Complementary ablation studies validate the efficacy of individual components constituting our proposed DRO approach.

In the second method, we introduce a novel perspective through a multi-view graph contrastive learning paradigm, which capitalizes on node-level local patterns and overarching graph-level representations to empower neural heuristics augmented with active search for solving Vehicle Routing Problems (VRPs). A comprehensive suite of experiments including synthetic instances and benchmark instances

from the TSPLib and CVRPLib repositories, which contains a range of diverse distributions, demonstrates the improvement in cross-distribution generalization performance facilitated by our Multi-View Graph Contrastive Learning (MVGCL) framework.

In the third method, we present an Ensemble-based Deep Reinforcement Learning (EL-DRL) methodology designed to solve vehicle routing problems under distribution shifts. This framework strategically enhances the adaptability of existing neural heuristics to address distributional shifts. By diversifying distinct reinforcement loss signals for each sub-policy and incorporating regularization terms to ensure the dispersion of approaches, our EL-DRL effectively leverages the unique strength of individual sub-policies when faced with instances characterized by diverse distributions. Empirical results confirm the high-quality solutions produced by our approach, not only across synthetic instances with varying distributions but also across heterogeneous instances drawn from real-world benchmarks. The effectiveness of the ensemble strategy is firmly established, highlighting its role in advancing more adaptable and resilient neural heuristics.

In conclusion, we approach the challenge of distribution shifts from diverse viewpoints and with varying models. With each step forward, we meticulously design new methods that progressively reduce the gap to the optimal solution. Our group Distributionally Robust Optimization (DRO) approach combined with Multi-View Graph Contrastive Learning (MVGCL) and Ensemble Learning-enhanced Deep Reinforcement Learning (EL-DRL) collectively achieves cutting-edge performance in their respective literature. Notably, EL-DRL stands out with exceptional generalization performance in addressing vehicle routing problems under distribution shift.

6.2 Future Work

All the heuristics we proposed to solve routing problems using neural techniques has the potential to handle more complex types of CoPs. First, we could expand the current method to deal with more constraints, so it considers both the constraints and the next best step to take. This augmentation could be feasibly realized through methodological paradigms such as reward shaping or hierarchical

Reinforcement Learning (RL). Additionaly, our methods, such as DRO and EL-DRP, are versatile and applicable to a range of other problems. These include Split Delivery Routing Problems (SDVRP), Orienteering Problems (OP), Prize Collecting TSP (PCTSP), Stochastic PCTSP (SPCTSP), and other non-routing problems.

Furthermore, even though we try to use simple ways to find solutions in this study, our method can also integrate more advanced methods. This encompasses popular methods such as simulated annealing, genetic algorithms, large neighborhood search, evolutionary algorithms and HGS. Notably, the generalizability of our methodology also extends to other pivotal COP domains, including but not limited to scheduling and bin packing. Subsequent research inquiries are geared towards a thorough exploration of these promising trajectories. Moreover, advanced reinforcement learning (RL) algorithms, such as Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C), can be employed to address these problems. However, existing research [125] demonstrates that directly applying PPO to vehicle routing problems does not yield substantial improvements. Therefore, when tackling Vehicle Routing Problems (VRPs), it is crucial to gain a deep understanding of the problem's inherent characteristics and tailor the RL algorithm accordingly.

In the future, we will explore more flexible distribution combinations and address larger-scale problems. Our focus will center on improving the inference efficiency of MVGCL and handling the effectiveness of our EL-DRL in larger instances. A significant reason for EL-DRL's degraded performance in such cases is its scalability depends on the sub-model, particularly the underperformance of POMO on larger instances. Additionally, due to the ensemble nature of EL-DRL, its training demands greater computational resources compared to a single model. Our future goals involve: 1) enhancing scalability by investigating decomposition methods like divide-and-conquer [122] for more efficient solutions to extensive instances; 2) increasing computational efficiency by introducing sparsity in ensemble learning [123], enabling training of more sub-models with reduced overhead; and 3) enabling interpretability [124] and reinforcing collaboration among sub-models in our approach.

List of Publications

Journal Papers

- **Yuan Jiang**, Zhiguang Cao, Jie Zhang. Learning to solve 3-D bin packing problem via deep reinforcement learning and constraint programming. IEEE transactions on cybernetics (**IEEE Trans. Cybern.**), 2021.

Conference Papers

- **Yuan Jiang**, Zhiguang Cao, Yaoxin Wu, Jie Zhang. Multi-View Graph Contrastive Learning for Solving Vehicle Routing Problems. 39th Conference on Uncertainty in Artificial Intelligence (**UAI, Spotlight**), 2023.
- **Yuan Jiang**, Yaoxin Wu, Zhiguang Cao, Jie Zhang. Learning to Solve Routing Problems via Distributionally Robust Optimization. 36th AAAI Conference on Artificial Intelligence (**AAAI, Oral**), 2022.
- **Yuan Jiang**, Zhiguang Cao, Jie Zhang. Solving 3D bin packing problem via multimodal deep reinforcement learning. Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (**AAMAS, Extended Abstract**), 2021.
- **Yuan Jiang**, Zhiguang Cao, Yaoxin Wu, Wen Song, Jie Zhang. Ensemble-based Deep Reinforcement Learning for Vehicle Routing Problems under Distribution Shift. 37th International Conference on Advances in Neural Information Processing Systems (**Submitted**), 2023.

Bibliography

[1] KORTE Bernhard and JENS Vygen. Combinatorial optimization: Theory and algorithms. *Springer, 5th Edition.*, 2012. 1

[2] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. 1

[3] Natalia Vesselinova, Rebecca Steinert, Daniel F Perez-Ramirez, and Magnus Boman. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access*, 2020. 1, 9

[4] 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. 1, 9

[5] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Proc. of NeurIPS*, 2018. 1

[6] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *the 7th International Conference on Learning Representations*, 2019. 2, 10, 23, 30, 31, 39, 45, 47, 50, 51

[7] Jingwen Li, Liang Xin, Zhiguang Cao, Andrew Lim, Wen Song, and Jie Zhang. Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[8] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Step-wise deep learning models for solving routing problems. *IEEE Transactions on Industrial Informatics*, 17(7):4861–4871, 2020.

[9] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 10

[10] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of 35th AAAI Conference on Artificial Intelligence*, 2021. 10, 13, 50

[11] Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[12] Liaing Xin, Wen Song, Zhiguang Cao, and Jie Zhang. 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, 2021. 1

[13] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015. 2, 10

[14] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. In *INFORMS Annual Meeting*, 2019. 2, 5, 10, 19, 22, 30, 34

[15] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017. 2

[16] Irwan Bello and Hieu Pham. Neural combinatorial optimization with reinforcement learning. In *the 5th International Conference on Learning Representations*, 2017. 2, 3, 10, 37, 38

[17] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020. 5, 6, 10, 19, 22, 30, 31, 32, 37, 39, 40, 45, 46, 48, 49, 50, 51, 53

[18] Minsu Kim, Jinkyoo Park, et al. Learning collaborative policies to solve np-hard routing problems. *Proc. of NeurIPS*, 2021. 2, 10, 37, 39, 47

[19] Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 2022. 2

[20] 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. In *Advances in Neural Information Processing Systems*, 2022. 2, 6, 11, 51, 55

[21] Aigerim Bogyrbayeva, Meraryslan Meraliyev, Taukekhan Mustakhov, and Bissenbay Dauletbayev. Learning to solve vehicle routing problems: A survey. *arXiv preprint arXiv:2205.02453*, 2022. 3

[22] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017. 3, 10

[23] Lu Duan, Haoyuan Hu, Yu Qian, Yu Gong, Xiaodong Zhang, Jiangwen Wei, and Yinghui Xu. A multi-task selected learning approach for solving 3d flexible bin packing problem. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1386–1394, 2019.

[24] Yuan Jiang, Zhiguang Cao, and Jie Zhang. Learning to solve 3-d bin packing problem via deep reinforcement learning and constraint programming. *IEEE transactions on cybernetics*, 2021. 3, 9

[25] André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *Proc. of ICLR*, 2022. 6, 32, 37, 38, 39, 43, 44

[26] Yuan Jiang, Yaoxin Wu, Zhiguang Cao, and Jie Zhang. Learning to solve routing problems via distributionally robust optimization. In *Proc. of AAAI*, 2022. 6, 39, 51

[27] 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*, pages 1–27, 2021. 9

[28] Luca Accorsi, Andrea Lodi, and Daniele Vigo. Guidelines for the computational testing of machine learning approaches to vehicle routing problems. *Operations Research Letters*, 50(2):229–234, 2022.

[29] Matthew Veres and Medhat Moussa. Deep learning for intelligent transportation systems: A survey of emerging trends. *IEEE Transactions on Intelligent transportation systems*, 21(8):3152–3168, 2019. 9

[30] Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. Learning to dispatch for job shop scheduling via deep reinforcement learning. In *the 34th Conference on Advances in Neural Information Processing Systems ((NeurIPS)*, volume 33, pages 1621–1632, 2020. 9, 45

[31] Fei Ni, Jianye Hao, Jiawen Lu, Xialiang Tong, Mingxuan Yuan, Jiahui Duan, Yi Ma, and Kun He. A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3441–3451, 2021.

[32] Runzhong Wang, Zhigang Hua, Gan Liu, Jiayi Zhang, Junchi Yan, Feng Qi, Shuang Yang, Jun Zhou, and Xiaokang Yang. A bi-level framework for learning to solve combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 34, 2021. 9

[33] José García, Eduardo Lalla-Ruiz, Stefan Voß, and Enrique López Droguett. Enhancing a machine learning binarization framework by perturbation operators: analysis on the multidimensional knapsack problem. *International Journal of Machine Learning and Cybernetics*, 11(9):1951–1970, 2020. 9

[34] Emir Demirović, Peter J Stuckey, James Bailey, Jeffrey Chan, Chris Leckie, Kotagiri Ramamohanarao, and Tias Guns. An investigation into prediction+ optimisation for the knapsack problem. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 241–257. Springer, 2019.

[35] Ole-Christoffer Granmo, B John Oommen, Svein Arild Myrer, and Morten Goodwin Olsen. Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):166–175, 2007. 9

[36] Hang Zhao, Yang Yu, and Kai Xu. Learning efficient online 3d bin packing on packing configuration trees. In *Proceedings of the 10th International Conference on Learning Representations*, 2021. 9

[37] Haoyuan Hu, Xiaodong Zhang, Xiaowei Yan, Longfei Wang, and Yinghui Xu. Solving a new 3d bin packing problem with deep reinforcement learning method. *arXiv preprint arXiv:1708.05930*, 2017. 9

[38] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021. 9

[39] Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, pages 6278–6289, 2019. 10, 45

[40] Hao Lu, Xingwen Zhang, and Shuang Yang. A learning-based iterative method for solving vehicle routing problems. In *International Conference on Learning Representations*, 2019. 10, 13

[41] André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. In *Proceedings of the 24th European Conference on Artificial Intelligence*, 2020. 10, 39, 51

[42] Paulo R d O Costa, Jason Rhuggenaath, Yingqian Zhang, and Alp Akcay. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Proc. of ACML*, 2020. 10

[43] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, 2018. 10

[44] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems*, 31, 2018. 10, 46

[45] Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proc. of AAAI*, 2021. 10, 34

[46] 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*, 2022. 10

[47] Simon Geisler, Johanna Sommer, Jan Schuchardt, Aleksandar Bojchevski, and Stephan Günnemann. Generalization of neural combinatorial solvers through the lens of adversarial robustness. In *Proc. of ICLR*, 2022. 10

[48] Zeyang Zhang, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning to solve travelling salesman problem with hardness-adaptive curriculum. In *Proc. of AAAI*, 2022. 10, 11, 29, 37, 39, 51

[49] Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto, and Percy Liang. Distributionally robust language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 4227–4237, 2019. 11, 17, 19

[50] Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. Wasserstein distributionally robust optimization: Theory and applications in machine learning. In *Operations Research & Management Science in the Age of Analytics*, pages 130–166. INFORMS, 2019. 11

[51] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pages 9229–9248, 2020. 11

[52] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289, 2019. 11

[53] Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013. 11

[54] John C Duchi, Tatsunori Hashimoto, and Hongseok Namkoong. Distributionally robust losses against mixture covariate shifts. *arXiv preprint arXiv:2007.13982*, 2019. 11, 18

[55] Hongseok Namkoong and John C Duchi. Variance-based regularization with convex objectives. In *Advances in Neural Information Processing Systems*, pages 2975–2984, 2017. 11, 19

[56] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. 11

[57] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, pages 2029–2037, 2018. 11

[58] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2019. 11, 19, 20, 22

[59] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. of CVPR*, 2020. 12, 33, 35, 36

[60] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Proc. of ECCV*, 2020. 31

[61] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 12, 31, 35

[62] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Proc. of NeurIPS*, 2013. 12

[63] John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. Declutr: Deep contrastive learning for unsupervised textual representations. In *Proc. of ACL*, 2021. 12, 31

[64] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proc. of KDD*, 2020. 12, 33

[65] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *Proc. of ICML*, 2020. 12

[66] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 12, 13

[67] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *Proc. of ICML*, 2021. 12

[68] Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. Autogcl: Automated graph contrastive learning via learnable view generators. In *Proc. of AAAI*, 2022. 12, 13

[69] Jiajun Zhou, Chenxuan Xie, Zhenyu Wen, Xiangyu Zhao, and Qi Xuan. Data augmentation on graphs: A survey. *arXiv preprint arXiv:2212.09970*, 2022. 12, 36

[70] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Proc. of NeurIPS*, 2020. 12, 33, 36

[71] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*, 2021. 13

[72] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Costa: Covariance-preserving feature augmentation for graph contrastive learning. In *Proc. of KDD*, 2022. 13

[73] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *knowledge discovery and data mining*, 2014. 13

[74] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. *the web conference*, 2015.

[75] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proc. of KDD*, 2016. 13

[76] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *Proc. of ICLR*, 2020. 13

[77] Mudasir A Ganaie, Minghui Hu, AK Malik, M Tanveer, and PN Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022. 13

[78] Yongquan Yang, Haijun Lv, and Ning Chen. A survey on ensemble learning under the era of deep learning. *Artificial Intelligence Review*, 56(6):5545–5589, 2023. 13

[79] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30:8008–8018, 2021. 13

[80] M. A. Ganaie, Minghui Hu, M. Tanveer, and P. N. Suganthan. Ensemble deep learning: A review. 2021. 13

[81] Gaon An, Seungyong Moon, Jang Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. 2021. 13

[82] Zhengyu Yang, Kan Ren, Xufang Luo, Weiqing Liu, Jiang Bian, Weinan Zhang, and Dongsheng Li. Deep ensemble policy learning, 2022. URL `https://openreview.net/forum?id=-7NOEQcD-xH`. 13

[83] E Ke Tang, Ponnuthurai N Suganthan, and Xin Yao. An analysis of diversity measures. *Machine learning*, 65:247–271, 2006. 14

[84] Ammar Mohammed and Rania Kora. A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University-Computer and Information Sciences*, 2023. 14

[85] Ling Liu, Wenqi Wei, Ka-Ho Chow, Margaret Loper, Emre Gursoy, Stacey Truex, and Yanzhao Wu. Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness. In *2019 IEEE 16th international conference on mobile ad hoc and sensor systems (MASS)*, pages 274–282. IEEE, 2019. 14

[86] Hassam Sheikh, Mariano Phielipp, and Ladislau Boloni. Maximizing ensemble diversity in deep reinforcement learning. In *International Conference on Learning Representations*, 2022. 14, 49

[87] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 20, 37, 48

[88] Shoma Miki and Hiroyuki Ebara. Solving traveling salesman problem with image-based classification. In *Proceedings of the 31st International Conference on Tools with Artificial Intelligence*, pages 1118–1123, 2019. 21

[89] Zhengxuan Ling, Xinyu Tao, Yu Zhang, and Xi Chen. Solving optimization problems through fully convolutional networks: An application to the traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–11, 2020. doi: 10.1109/TSMC.2020.2969317. 21

[90] Jakob Bossek, Pascal Kerschke, Aneta Neumann, Markus Wagner, Frank Neumann, and Heike Trautmann. 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*, pages 58–71, 2019. 22, 40, 52

[91] Kangfei Zhao, Shengcai Liu, Yu Rong, and Jeffrey Xu Yu. Leveraging tsp solver complementarity via deep learning. *arXiv e-prints*, pages arXiv–2006, 2020. 22

[92] David Applegate, Ribert Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver. *URL http://www.math.uwaterloo.ca/tsp/concorde*, 2006. 22, 39, 50

[93] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000. 22

[94] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019. 23

[95] Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991. 27, 42, 50, 55

[96] 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. 27

[97] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proc. of EMNLP*, 2021. 31

[98] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proc. of CVPR*, 2018. 33

[99] Zhongzhi Zhang, Tong Shan, and Guanrong Chen. Random walks on weighted networks. *Physical Review E*, 2013. 34

[100] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Proc. of ICDM*, 2006. 35

[101] Silvan Sievers, Michael Katz, Shirin Sohrabi, Horst Samulowitz, and Patrick Ferber. Deep learning for cost-optimal planning: Task-dependent planner selection. In *Proc. of AAAI*, 2019. 36

[102] Kangfei Zhao, Shengcai Liu, Jeffrey Xu Yu, and Yu Rong. Towards feature-free tsp solver selection: A deep learning approach. In *Proc. of IJCNN*, 2021. 36

[103] Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap neighborhood. *Computers & Operations Research*, 2022. 39, 51

[104] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 40

[105] Eduardo Queiroga, Ruslan Sadykov, Eduardo Uchoa, and Thibaut Vidal. 10,000 optimal cvrp solutions for testing machine learning based heuristics. In *Proc. of AAAI*, 2021. 42, 50, 55

[106] Mengyuan Lee, Seyyedali Hosseinalipour, Christopher Greg Brinton, Guanding Yu, and Huaiyu Dai. A fast graph neural network-based method for winner determination in multi-unit combinatorial auctions. *IEEE Transactions on Cloud Computing*, 2020. 45

[107] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[108] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 7087–7094, 2018.

[109] André Hottung, Bhanu Bhandari, and Kevin Tierney. Learning a latent search space for routing problems using variational autoencoders. In *the 8th International Conference on Learning Representations (ICLR)*, 2020. 45

[110] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017. 46

[111] Oren Peer, Chen Tessler, Nadav Merlis, and Ron Meir. Ensemble bootstrapping for q-learning. In *International Conference on Machine Learning*, pages 8454–8463. PMLR, 2021.

[112] Litian Liang, Yaosheng Xu, Stephen McAleer, Dailin Hu, Alexander Ihler, Pieter Abbeel, and Roy Fox. Reducing variance in temporal-difference value estimation via ensemble of deep networks. In *International Conference on Machine Learning*, pages 13285–13301. PMLR, 2022. 46

[113] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 46

[114] Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3243–3250, 2020. 46

[115] 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. 46

[116] Siobhán O'mahony and Fabrizio Ferraro. The emergence of governance in an open source community. *Academy of Management Journal*, 50(5):1079–1106, 2007. 48

[117] Bradley Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982. 48

[118] J Johnston. H. theil. economics and information theory, 1969. 49

[119] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011. 50

[120] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 2017. 51, 53

[121] G Brown. Diversity in neural network ensembles,[ph. d. thesis]. *University of Birmingham*, 2004. 56

[122] Sirui Li, Zhongxia Yan, and Cathy Wu. Learning to delegate for large-scale vehicle routing. *Proc. of NeurIPS*, 2021. 57, 61

[123] Li Zhang and Wei-Da Zhou. Sparse ensembles using weighted combination methods based on linear programming. *Pattern Recognition*, 44(1):97–106, 2011. 57, 61

[124] Chia-Hsiu Chen, Kenichi Tanaka, Masaaki Kotera, and Kimito Funatsu. Comparison and improvement of the predictability and interpretability with ensemble learning models in qspr applications. *Journal of cheminformatics*, 12:1–16, 2020. 58, 61

[125] Han Lu, Zenan Li, Runzhong Wang, Qibing Ren, Xijun Li, Mingxuan Yuan, Jia Zeng, Xiaokang Yang, and Junchi Yan. ROCO: A general framework for evaluating robustness of combinatorial optimization solvers on graphs. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=2r6YMqz4Mml`. 61