Multi-View Graph Contrastive Learning for Solving Vehicle Routing Problems

Yuan Jiang[1] **Zhiguang Cao**[2] **Yaoxin Wu**[3*] **Jie Zhang**[1]

[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]School of Computing and Information Systems, Singapore Management University, Singapore
[3]Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Netherlands
[*]Corresponding Author

Abstract

Recently, neural heuristics based on deep learning have reported encouraging results for solving vehicle routing problems (VRPs), especially on independent and identically distributed (i.i.d.) instances, e.g. *uniform*. However, in the presence of a distribution shift for the testing instances, their performance becomes considerably inferior. In this paper, we propose a multi-view graph contrastive learning (MVGCL) approach to enhance the generalization across different distributions, which exploits a graph pattern learner in a self-supervised fashion to facilitate a neural heuristic equipped with an active search scheme. Specifically, our MVGCL first leverages graph contrastive learning to extract transferable patterns from VRP graphs to attain the generalizable multi-view (i.e. node and graph) representation. Then it adopts the learnt node embedding and graph embedding to assist the neural heuristic and the active search (during inference) for route construction, respectively. Extensive experiments on randomly generated VRP instances of various distributions, and the ones from TSPLib and CVRPLib show that our MVGCL is superior to the baselines in boosting the cross-distribution generalization performance.

1 INTRODUCTION

Vehicle routing problem (VRP) is essentially a combinatorial optimization problem (COP) with many important real-world applications, especially in logistics [Bernhard and Vygen, 2012]. In reality, it occurs more than often that vehicle routing tasks are repeatedly carried out which share similar problem structures but only differ in data. For example, a logistic company may dispatch a fleet of trucks to pick up or deliver packages for customers in the same city on a daily basis, with only discrepancies in customer locations and/or demands. However, conventional heuristic methods always treat each of those tasks independently, which may yield limited computation efficiency and/or solution quality. Hence, developing *neural* heuristics based on deep learning has become a sought-after alternative for solving VRPs, which aim to improve the performance by exploiting the underlying patterns in the instances [Bengio et al., 2021].

The early neural heuristics for VRPs primarily fall into the supervised category, which requires the optimal solution as supervised labels [Vinyals et al., 2015, Joshi et al., 2019]. The resulting performance highly relies on the quality of the labels, rendering the supervised methods are less favourable since it is computationally expensive to attain optimal solutions due to the NP-hardness. Moreover, it is also difficult for supervised methods to generalize to problem sizes different from the training ones. In contrast, the neural heuristics based on (deep) reinforcement learning only need a reward (e.g. the current tour length) rather than the optimal solution at each decision step, to indicate whether a move or a selection is favourable or not [Bello et al., 2017, Kool et al., 2019, Kwon et al., 2020, Kim et al., 2021]. Meanwhile, it is also relatively easier for them to generalize to different problem sizes than the supervised ones.

Although the neural heuristics have reported many encouraging results for VRPs, most of their underlying models are trained and evaluated on independent and identically distributed (i.i.d.) instances with respect to the node locations, especially the *uniform* distribution. An ideal neural heuristic should be able to generalize to various distributions, since the real-world instances may follow different and sometimes even unknown distributions. Unfortunately, directly applying existing neural heuristics trained on the uniform distribution to instances of other distributions will result in considerably inferior solutions [Geisler et al., 2022, Zhang et al., 2022b], which may hinder their applications. On the other hand, some preliminary studies have been conducted to alleviate this generalization issue, which leverage group distributionally robust optimization (DRO) [Jiang

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

et al., 2022] or adaptive hardness assisted curriculum learning (HAC) [Zhang et al., 2022b] to train the model. However, the former needs to label typical and atypical instances, and the latter is mainly extended to Gaussian distribution only.

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) [Chen et al., 2020a, Wu et al., 2021b, Hudson et al., 2022], we postulate that transferable structural patterns across diverse graphs could be helpful to improve the generalization against different distributions [Qiu et al., 2020, Leskovec et al., 2005]. 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) [He et al., 2020, Chen et al., 2020b, Giorgi et al., 2021, Gao et al., 2021, Hassani and Khasahmadi, 2020, You et al., 2021] have testified that pre-training an encoder network in a contrastive learning manner can produce more informative and transferable representation for downstream tasks.

With the above principle, in this paper, 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 [Kwon et al., 2020]) and its active search scheme [Hottung et al., 2022] (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., the travelling salesman problem (TSP) and capacitated vehicle routing problem (CVRP). Results on randomly generated instances and benchmark ones (i.e. TSPLib and CVRPLib) verified its effectiveness.

2 RELATED WORKS

2.1 NEURAL HEURISTICS FOR VRPS

In recent years, the neural heuristics based on deep (reinforcement) learning for routing problems have been extensively explored. The pointer network (PtrNet) [Vinyals et al., 2015], as the first modern deep architecture for VRPs is essentially developed from the encoder-decoder-based sequence-to-sequence model for NLP. Bello et al. [2017] propose to train PtrNet with reinforcement learning since ground-truth labels are computationally expensive. Kool et al. [2019] further boost the performance by introducing a self-attention encoder [Vaswani et al., 2017] and an attentive decoder, which stands as the well-known attention model (AM). Kwon et al. [2020] propose the POMO model on top of AM by augmenting the input instances and starting the inference from multiple nodes. Hottung et al. [2022] conceive three different strategies to integrate the active search with POMO during the inference phase, which deliver noticeably superior performance. Differently, a concurrent line of *improvement* methods [Chen and Tian, 2019, Lu et al., 2019, d O Costa et al., 2020, Wu et al., 2021b, Kim et al., 2021] emphasize improving an initial but complete solution via iterative local operations. Among them, Kim et al. [2021] 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, GNNs or its variants such as graph convolutional networks (GCNs) and graph attention networks (GATs) are also exploited on VRP graphs [Khalil et al., 2017, Deudon et al., 2018, Joshi et al., 2019]. Besides, Li et al. [2018] propose guided tree search by leveraging GCN embeddings. Fu et al. [2021] 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 [Kwon et al., 2020, Kool et al., 2022]. However, Geisler et al. [2022] and Zhang et al. [2022b] 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, Jiang et al. [2022] exploit group distributionally robust optimization (DRO) to train deep models (i.e. POMO and GCN) across multiple distributions, which needs to label typical and atypical instances. Zhang et al. [2022b] propose a curriculum learning-based AM trained on instances of different hardness. Those instances are generated by a hardness-adaptive generator with mixed-Gaussian distribution, which limits its generalization to more others.

2.2 GRAPH CONTRASTIVE LEARNING

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 [He et al., 2020, Tian et al., 2020, Chen et al., 2020b] and NLP [Mikolov et al., 2013, Giorgi et al., 2021].

To tackle the graphic data, graph contrastive learning (GCL) has been proposed [Qiu et al., 2020, Hassani and Khasahmadi, 2020, Liu et al., 2022], and a series of augmentation techniques to generate contrastive samples based on the original graph have also been accordingly developed [You et al., 2021, Yin et al., 2022, Zhou et al., 2022], such as attribute removing, edge adding/masking, and subgraph/graph diffusion. Among them, Qiu et al. [2020] 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 [2020] contrastively learn embeddings from the first-order neighbours and graph diffusion. GraphCL [You et al., 2020] 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 [You et al., 2021]. Similarly, Yin et al. [2022] 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 [You et al., 2020]; 2) VRP graphs are fully-connected, augmentations like node dropping or edge perturbation [Wu et al., 2021a, Liu et al., 2022] may violate the connectedness. Worse still, the original graph distribution could be distorted by altering its structure. Although recent works [Yin et al., 2022, Zhang et al., 2022a] attempt to preserve graph class labels during augmentation, it is impractical to acquire all distribution labels for VRPs. Furthermore, unlike traditional network embeddings [Perozzi et al., 2014, Tang et al., 2015, Grover and Leskovec, 2016] or recent works that pre-train GNNs with attributed graphs and then directly apply them to instances of the same domain [Hu et al., 2020], 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.

3 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.

3.1 PRELIMINARY

We consider the two classic VRP variants, i.e., travelling salesman problem (TSP) and capacitated vehicle routing problem (CVRP) in the 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.

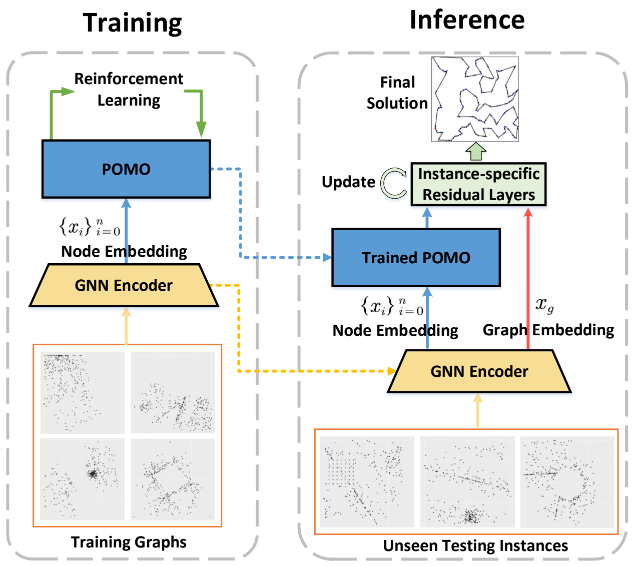

Figure 1: The illustration of our MVGCL, where the multi-view embeddings from the pre-trained GNN will be used to facilitate the subsequent training and inference.

3.2 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 [Kool et al., 2019, Joshi et al., 2019, Kwon et al., 2020] for VRPs did not explicitly consider such matter and thus the underlying models trained on one distribution often yield inferior cross-distribution generalization.

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 [Tian et al., 2020, Chen et al., 2020b] or NLP [Giorgi et al., 2021, Gao et al., 2021], 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

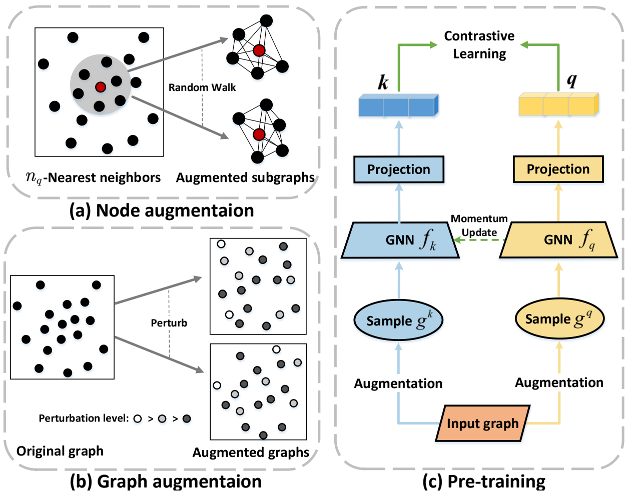

Figure 2: The framework of our node-level and graph-level contrastive learning, where the node augmentation in (a) and graph augmentation in (b) will share the same pre-training paradigm in (c) to attain the GNN f_q (f_k will be discarded).

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 [Kool et al., 2019, Kwon et al., 2020], explicitly (pre-)training over such 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 [Kwon et al., 2020]) trained with reinforcement learning, and the learnt graph embedding is employed to facilitate the active search [Hottung et al., 2022] equipped to the neural heuristic during inference. The overall framework of our MVGCL is depicted in Figure 1.

3.3 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 [Wu et al., 2018, He et al., 2020, Qiu et al., 2020], 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,

$$\mathcal{L}_q = -\log \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_+/\tau\right)}{\sum_{i=0}^{K} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i/\tau\right)}, \tag{1}$$

where τ is a hyper-parameter of temperature that regulates the weights of penalties on negative samples [Wu et al., 2018]. 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.

3.3.1 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 [You et al., 2020, Qiu et al., 2020]. 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 [Joshi et al., 2019, Fu et al., 2021], 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 did 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 2 (a), we first extract a n_q-nearest-neighbours subgraph for a node and then apply the multi-hop random walk (MHRW) [Zhang et al., 2013] 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. The details of our adapted MHRW can be found in Appendix A.

Meanwhile, we adopt the Momentum Contrast (MoCo) [He et al., 2020, Chen et al., 2020b] 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 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,

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q. \tag{2}$$

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 key (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. (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.

3.3.2 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 [Sievers et al., 2019, Zhao et al., 2021], 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 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 [You et al., 2020, Zhou et al., 2022].

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,

$$\mathbf{v}_i' = \mathbf{v}_i + \eta \cdot \Delta\mathbf{v}_i; \quad \Delta\mathbf{v}_i \sim \mathcal{U}\left[-\eta, \eta\right]^2, \tag{3}$$

where $\Delta\mathbf{v}_i$ is uniformly sampled in a square with length 2η. In this way, most (about $p_1 * n$) 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 $p_2 * n$) 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 $p_3 * n$) 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 2 (c)). During the training of MoCo [He et al., 2020], we feed those pairs into the encoders. Finally, GNN f_q learns to produce representations for similar distributions close 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 [Bello et al., 2017, Hottung et al., 2022], 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.

3.4 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.

3.4.1 Policy optimization with node embedding

In the training phase, we use Policy Optimization with Multiple Optima (POMO) [Kwon et al., 2020] 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 [Kwon et al., 2020, Kim et al., 2021, Zhang et al., 2022b], 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 [Williams, 1992], as did in the original POMO.

3.4.2 Active search with graph embedding

In the inference phase, to better generalize the trained model on unseen instances, we exploit active search [Bello et al., 2017] to dynamically adjust the parameters for each target instance. Since updating all parameters of the model is expensive and impractical during inference, we adopt a similar technique as [Hottung et al., 2022], which adds instance-specific residual layers before the output layer of the decoder in POMO. And we only update the parameters of these residual layers. 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 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

$$\begin{aligned} h_l &= \hat{h} + \left(\left(\mathrm{ReLu}\left(\hat{h}W_l^1 + b_l^1\right)W_l^2 + b_l^2\right)\right), \\ \hat{h} &= \begin{cases} [h_n, x_g] & , l = 1, \\ h_{l-1} & , l > 1, \end{cases} \end{aligned} \tag{4}$$

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. For more details of POMO and active search, please refer to Appendix B.

4 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.

4.1 EXPERIMENTAL SETTINGS

Baselines. For TSP, we compare with the exact solver Concorde [Applegate et al., 2006] and representative neural heuristics including Attention Model (AM) [Kool et al., 2019], Policy Optimization with Multiple Optima (POMO) [Kwon et al., 2020], Learning Collaborative Policies (LCP) [Kim et al., 2021], Distributionally Robust Optimization with POMO (DROP) [Jiang et al., 2022], and Hardness-Adaptive Curriculum (HAC) [Zhang et al., 2022b], Efficient Active Search (EAS) [Hottung et al., 2022]. 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 [Vidal, 2022] as a conventional baseline, which reported superior performance to LKH3 [Hottung and Tierney, 2020]. 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 for ablation study.

Implementation. We adopt the Graph Isomorphism Network (GIN) [Xu et al., 2018] 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 [Kwon et al., 2020], except that the batch size is reduced from 64 to 56 for CVRP100 due to the memory limit. 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. The details of our implementation, including hardware, hyperparameters and network architecture, are presented in Appendix C.

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 [Bossek et al., 2019], 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 best performance. We specify the last applied mutation as the class label for

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

Table 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

each instance, since the training of DROP needs this label.

4.2 GENERALIZATION ON TSP

In Table 1, we display the average values of tour lengths, gaps 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, thus induce longer runtime.

4.3 GENERALIZATION ON CVRP

We display the results for CVRP50 and CVRP100 in Table 2. As aforementioned, it is much harder to find the optimal solution for CVRP, thus 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. Furthermore, we also present results on TSP and CVRP instances of uniform distribution in Appendix D. To sum up, our method attains higher-quality solutions over most of the distributions.

4.4 RESULTS ON BENCHMARKS

We continue to evaluate our MVGCL on public benchmark datasets, i.e., TSPLib [Reinelt, 1991] and CVRPLib [Queiroga et al., 2021], 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 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 [Hottung et al., 2022], we directly use the models trained on TSP100 and CVRP100 (with mixed distributions) to solve those instances.

The upper half of Table 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 simi-

Table 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 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

Table 4: Ablation studies on MVGCL

	Component			TSP100	
Name	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%**

lar 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 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.

4.5 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, 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) [Hottung et al., 2022] 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.

5 CONCLUSIONS AND FUTURE WORKS

In this paper, 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. In future, we plan to further improve the inference efficiency of our MVGCL.

References

David Applegate, Robert Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver, 2006.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *Proc. of ICLR*, 2017.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 2021.

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

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*, 2019.

Mingxiang Chen, Lei Gao, Qichang Chen, and Zhixin Liu. Dynamic partial removal: A neural network heuristic for large neighborhood search. *arXiv preprint arXiv:2005.09330*, 2020a.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. *Proc. of NeurIPS*, 2019.

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.

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.

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.

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

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.

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

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

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

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. of CVPR*, 2020.

André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. In *Proc. of ECAI*, 2020.

André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *Proc. of ICLR*, 2022.

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.

Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. In *Proc. of ICLR*, 2022.

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

Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. In *INFORMS Annual Meeting*, 2019.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Proc. of NeurIPS*, 2017.

Minsu Kim, Jinkyoo Park, et al. Learning collaborative policies to solve np-hard routing problems. *Proc. of NeurIPS*, 2021.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *Proc. of ICLR*, 2019.

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.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Proc. of NeurIPS*, 2020.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. *knowledge discovery and data mining*, 2005.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. *Proc. of NeurIPS*, 2018.

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.

Hao Lu, Xingwen Zhang, and Shuang Yang. A learning-based iterative method for solving vehicle routing problems. In *Proc. of ICLR*, 2019.

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.

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

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.

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.

Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 1991.

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.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proc. of NeurIPS*, 2017.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proc. of NeurIPS*, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

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*, 2021a.

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*, 2021b.

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

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

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.

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

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

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

Zeyang Zhang, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning to solve travelling salesman problem with hardness-adaptive curriculum. In *Proc. of AAAI*, 2022b.

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

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.

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