Collaborative Deep Reinforcement Learning for Solving Multi-Objective Vehicle Routing Problems

Yaoxin Wu*
Eindhoven University of Technology
Eindhoven, Netherlands
wyxacc@hotmail.com

Mingfeng Fan*
National University of Singapore
Singapore, Republic of Singapore
mingfan2001@gmail.com

Zhiguang Cao
Singapore Management University
Singapore, Republic of Singapore
zhiguangcao@outlook.com

Ruobin Gao
Nanyang Technological University
Singapore, Republic of Singapore
ruobin.gao@ntu.edu.sg

Yaqing Hou†
Dalian University of Technology
Dalian, China
houyq@dlut.edu.cn

Guillaume Sartoretti
National University of Singapore
Singapore, Republic of Singapore
guillaume.sartoretti@nus.edu.sg

ABSTRACT

Existing deep reinforcement learning (DRL) methods for multi-objective vehicle routing problems (MOVRPs) typically decompose an MOVRP into subproblems with respective preferences and then train policies to solve corresponding subproblems. However, such a paradigm is still less effective in tackling the intricate interactions among subproblems, thus holding back the quality of the Pareto solutions. To counteract this limitation, we introduce a collaborative deep reinforcement learning method. We first propose a preference-based attention network (PAN) that allows the DRL agents to reason out solutions to subproblems in parallel, where a shared encoder learns the instance embedding and a decoder is tailored for each agent by preference intervention to construct respective solutions. Then, we design a collaborative active search (CAS) to further improve the solution quality, which updates only a part of the decoder parameters per instance during inference. In the CAS process, we also explicitly foster the interactions of neighboring DRL agents by imitation learning, empowering them to exchange insights of elite solutions to similar subproblems. Extensive results on random and benchmark instances verified the efficacy of PAN and CAS, which is particularly pronounced on the configurations (i.e., problem sizes or node distributions) beyond the training ones. Our code is available at https://github.com/marmotlab/PAN-CAS.

KEYWORDS

Multi-objective vehicle routing problems; Deep reinforcement learning; Attention network; Collaborative active search.

ACM Reference Format:
Yaoxin Wu, Mingfeng Fan, Zhiguang Cao, Ruobin Gao, Yaqing Hou, and Guillaume Sartoretti. 2024. Collaborative Deep Reinforcement Learning for Solving Multi-Objective Vehicle Routing Problems. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024,* IFAAMAS, 10 pages.

*Both authors contributed equally to the paper
†Corresponding authors

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

Multi-objective vehicle routing problems (MOVRPs), developed from single-objective VRPs (SOVRPs), are a class of VRPs characterized by a group of conflicting objectives to optimize. MOVRPs are integral to logistics, traffic management, urban planning, etc. [17], drawing significant interest from both industry and academia [51]. While SOVRPs are generally classified as NP-hard, MOVRPs present even greater complexity, as they aim to identify Pareto optimal solutions that offer different trade-offs between conflicting objectives. These solutions are significant as they reflect varying preferences for disparate criteria (i.e., objectives) and can be instrumental in decision-making processes to meet the diverse requirements of clients. However, exactly searching all Pareto optimal solutions proves to be impractically expensive, especially when dealing with large problem sizes or numerous solutions [10, 12]. As more practical alternatives, a variety of heuristics have been proposed, aimed at seeking a group of approximate Pareto solutions in manageable time frames. However, classic heuristics are often manually designed and delicately tuned to cater to specific MOVRPs [17], presenting a limitation that hinders the performance and applications to broader VRP variants. Therefore, automated learning of heuristics in a data-driven manner is highly desired, which is supposed to attenuate the above defects.

The past few years have witnessed surging research endeavors aimed at solving VRPs using deep reinforcement learning (DRL). Following a plethora of studies on SOVRPs [1, 3], deep models have been adapted and extended for MOVRPs. Typically, these models scalarize an MOVRP into multiple SOVRPs with respective preference vectors and then train DRL agents to solve these SOVRPs to obtain approximate (Pareto) solutions. In early attempts, a neural architecture such as Pointer Net or Attention Model [20, 42] for tackling a single objective is repeatedly trained for each SOVRP with transfer learning, which resulted in high training overhead and the impracticality of maintaining numerous neural networks. Recent endeavors have pivoted towards training a single neural architecture capable of solving all associated SOVRPs, incorporating an additional sub-network to process preferences and adjust the neural network to learn tailored policies for each specific SOVRP. Such a one-to-all training paradigm has shown promising results empirically, outperforming the earlier one-to-one paradigm and some traditional heuristics. However, the current one-to-all paradigm predominantly relies on simple interventions achieved by processing preferences through fully connected layers. This could potentially lead to undesirable proportionality dependency of the network parameters on preferences [34], thereby holding back the network capacity to derive more effective policies. Moreover, the existing DRL methods are struggling to address the interactions between subproblems, which could significantly enhance the overall quality of Pareto solutions [52], if properly leveraged.

To overcome the above issues, this paper introduces a collaborative deep reinforcement learning method to solve MOVRPs. Given an MOVRP instance, we decompose (scalarize) it into subproblems (SOVRPs) with respective preferences, and train DRL agents to reason out solutions to corresponding SOVRPs. To this end, we propose a preference-based attention network (PAN) to parameterize the policies in an encoding-decoding manner. The encoder in PAN is shared by DRL agents to learn the instance embedding, and the decoder is tailored for each respective agent by preference intervention to construct solutions in parallel. In particular, we project preferences into query, key, and value matrices in attentions of the decoder, which are added with their preference-agnostic but trainable counterparts of query, key, and value. On top of PAN, we also propose a collaborative active search (CAS) to further improve the quality of Pareto solutions during inference. CAS fine-tunes the policies of DRL agents per instance by updating only a subset of the decoder parameters. Throughout this process, we also explicitly promote the interactions between neighboring agents addressing similar subproblems with similar preferences. Each agent is allowed to perceive the respective solutions of its neighbors for seeking and imitating the most elite one to its own subproblem. In doing so, the agents are able to efficiently exchange their insights of elite solutions to similar subproblems. In summary, this paper contributes to the DRL for MOVRPs community in the following aspects:

- We propose a collaborative DRL method to solve MOVRPs and exploit the encoder-decoder structured PAN for training policies to construct Pareto solutions in parallel.
- We introduce a hybrid intervention to the decoder in the PAN, leading to both the preference-based and preference-agnostic parameters in attentions for more effective policies. We also design CAS to further promote the solution quality, where DRL agents exchange insights of elite solutions by imitation learning.
- We evaluate the proposed approach on multi-objective traveling salesman problem (MOTSP) and multi-objective capacitated vehicle routing problem (MOCVRP). Results on synthetic and benchmark instances justified the efficacy of PAN and CAS, where our approach outperformed traditional heuristics and DRL baselines, especially on the configurations (i.e., problem sizes or node distributions) that are beyond the training ones.

2 RELATED WORK

Traditional Heuristics for MOVRPs. Solving MOVRPs is markedly more intractable than SOVRPs, such as the NP-hard TSP and CVRP [12]. The formidable computational complexity associated with MOVRPs often precludes the practical application of exact methods, especially in cases involving large problem sizes or a plethora of (Pareto) solutions [14, 46]. Therefore, the research focus has largely shifted towards heuristics designed to identify approximate Pareto solutions within acceptable timeframes [17, 51]. Among them, multi-objective evolutionary algorithms (MOEAs) have emerged as a prominent strategy, including the dominance-based MOEAs [8, 9, 37] and decomposition-based MOEAs [11, 18, 52]. However, the design of an MOEA often relies on laborious hand-engineering, which may impede the algorithmic performance. This often involves the predefinition of numerous algorithm components and exhaustive tuning of various reproduction, mutation, and selection operators, along with their associated hyperparameters [41, 48, 50]. Efforts have been made to attenuate the manual workload by automating some specific components, but massive tuning remains a prerequisite to discovering a reasonably good combination [24, 27, 40]. Furthermore, most heuristics are narrowly focused, often specialized and optimized for particular MOVRP like either MOTSP or MOCVRP [5, 13, 29, 32, 35, 39], which limits their adaptability to other VRP variants.

Deep Models for MOVRPs. Deep learning has demonstrated notable success in solving SOVRPs, especially the TSP and CVRP [2, 6, 15, 19, 23, 33, 47, 55], with comprehensive reviews available in [1, 36]. The prevailing approach in this line of literature harnesses the encoder-decoder structured neural architectures, exemplified by the models like Pointer Net [42] and Attention Model [20, 21]. Inspired by the works on SOVRPs, there have been endeavors to adapt deep models for MOVRPs, which utilize these neural architectures to solve a series of SOVRPs derived from the scalarized MOVRP. In particular, Wu et al. [45] and Li et al. [25] employ individual models, trained via RL algorithms, to solve each SOVRP. They transfer parameters among models for warm starts so as to improve the training convergence. Similarly, Shao et al. [38] and Zhang et al. [53] apply Pointer Net and Attention Model to tackle scalarized SOVRPs, respectively, and optimize the neural architecture parameters by evolutionary strategies. Zhang et al. [54] further introduces a meta-DRL procedure to enhance the knowledge transfer between subproblems. However, it necessitates considerable effort to fine-tune the meta-model for each SOVRP.

In the methods described above, each SOVRP is associated with an individually trained or fine-tuned neural network. Such a one-to-one paradigm often induces high training overhead and requires additional resources to maintain a set of neural networks. To mitigate the inconveniences, Lin et al. [28] presents a DRL method for MOVRPs, which achieves state-of-the-art performance among deep models and inspires subsequent research [26, 49]. These methods, characterized by a one-to-many paradigm, incorporate a sub-network to assimilate preference embeddings. These embeddings are then transformed into parameters within the decoder of Attention Model, giving rise to respective routing policies tailored for each SOVRP. Nevertheless, the current one-to-many paradigm solely embeds preferences via simple fully connected networks, which could potentially lead to inferior parameters in the decoder and inhibit the learning of more effective policies.

3 PRELIMINARIES

3.1 MOVRP Description

Typically, a VRP instance can be defined on a graph $\mathcal{G}$ with the node set $\mathcal{V} = \{1, 2, \cdots, n\}$, where each node $i \in \mathcal{V}$ is featured by o_i.

The solution to a VRP instance is a tour $\pi = (\pi_1, \pi_2, \cdots, \pi_T)$, i.e., a node sequence of length T, with $\pi_j \in \mathcal{V}$. A solution π is feasible only if it meets the constraints for the VRP. Accordingly, an MOVRP with κ objectives are formally defined as:

$$\min_{\pi \in \mathcal{X}} F(\pi) = (f_1(\pi), f_2(\pi), \cdots, f_\kappa(\pi)), \tag{1}$$

where $\mathcal{X}$ comprises all feasible solutions to the MOVRP, and $F(\pi)$ is a κ-dimensional vector that includes κ objective values of the solution π. Given the conflict, there is no single solution for achieving optimality for every objective. Instead, Pareto solutions are often pursued to achieve different trade-offs (i.e., preferences) among objectives.

Definition 1 (Pareto Dominance). A solution $\pi \in \mathcal{X}$ dominates another solution $\pi' \in \mathcal{X}$ (i.e., $\pi \prec \pi'$), if and only if $f_i(\pi) \leq f_i(\pi'), \forall i \in \{1, \cdots, \kappa\}$ and $F(\pi) \neq F(\pi')$.

Definition 2 (Pareto Optimality). A solution $\pi^* \in \mathcal{X}$ is Pareto optimal if it is not dominated by any other solution. Accordingly, the Pareto set is defined as all Pareto optimal solutions, i.e., $\mathcal{P} = \left\{\pi^* \in \mathcal{X} | \nexists\, \pi' \in \mathcal{X} : \pi' \prec \pi\right\}$. The Pareto front is defined as images of Pareto optimal solutions in the objective space, i.e., $\mathcal{F} = \{F(\pi) | \pi \in \mathcal{P}\}$.

Since solving a SOVRP optimally (e.g., TSP and CVRP) is already NP-hard, MOVRPs are markedly more intractable with the aim of attaining Pareto optimal solutions, the quantity of which often exponentially expands along with the problem size (i.e., the number of nodes). Therefore, MOEAs are often adopted to compute approximate Pareto solutions. Among them, the MOEAs based on decomposition (MOEA/Ds) gain the solutions by solving SOVRPs decomposed from an MOVRP, which inspires current DRL methods.

3.2 MOEA/D Framework

The vanilla MOEA/D first utilizes decomposition techniques to scalarize an MOVRP into N subproblems (i.e., SOVRPs) with a set of uniformly distributed preferences $\lambda_1, \lambda_2, \cdots, \lambda_N$, each of which satisfies $\lambda_i = (\lambda_i^1, \cdots, \lambda_i^\kappa)^\top, \forall\, \lambda_i^j \geq 0$ and $\sum_{j=1}^{\kappa} \lambda_i^j = 1$.

3.2.1 Decomposition. In MOEA/Ds, the major decomposition techniques include weighted-sum, Tchebycheff, and penalty-based boundary intersection (PBI) approaches [31, 52], respectively.

Weighted-sum Approach. Given an MOVRP, the ith subproblem (i.e., SOVRP) is defined with the ith preference λ_i, such that,

$$\min \quad g_w(\pi|\lambda_i) = \sum\nolimits_{j=1}^{\kappa} \lambda_i^j f_j(\pi), \ \pi \in \mathcal{X} \tag{2}$$

Tchebycheff Approach. It minimizes the maximal distance between objectives and the ideal reference point, such that,

$$\min \quad g_t(\pi|\lambda_i, z^*) = \max_{1 \leq j \leq \kappa} \left\{\lambda_i^j |f_j(\pi) - z_j^*|\right\}, \ \pi \in \mathcal{X} \tag{3}$$

where $z^* = (z_1^*, \cdots, z_\kappa^*)^\top$ signifies the ideal reference point with $z_j^* \leq \min\left\{f_j(\pi) | \pi \in \mathcal{X}\right\}$. Unlike the weighted-sum approach, which is limited to convex Pareto fronts (PFs), the Tchebycheff approach is also effective for nonconvex PFs. It also guarantees that the optimal solution in Eq. (3) under a specific (but unknown) preference λ_i could be a Pareto optimal solution [7].

Algorithm 1 MOEA/D Procedure

```
Require: The number of subproblems $N$; the number of objectives $m$; the neighbourhood size $N_s$; the ideal point $z^*$.
1: Initialize a population of solutions $P = \{\pi^1, \cdots, \pi^N\}$, a set of $N$ uniform preferences $\lambda_1, \lambda_2, \cdots, \lambda_N$ and their neighborhoods. Assign the solution $\pi^i$ to the preference $\lambda_i$, $i \in \{1, \cdots, N\}$.
2: while not satisfy the stopping criteria do
3:     for $i = 1, \cdots, N$ do
4:         Randomly select a required number of mating parents from $\lambda_i$'s neighborhood, denoted as $\psi^i$.
5:         Using crossover and mutation operations to reproduce offspring $\pi^{i,c}$ of $\pi^i$.
6:         Update the solutions linked to subproblems within $\psi^i$ using $\pi^{i,c}$ if $g_t(\pi^{i,c}|\lambda_k, z^*) < g_t(\pi^k|\lambda_k, z^*)$, where $\lambda_k \in \psi^i$ and $\pi^k$ is the current solution associated with $\lambda_k$.
7:     end for
8: end while
9: return Final population.
```

PBI Approach. This approach formulates the ith subproblem of an MOVRP as follows,

$$\begin{aligned} \min \quad & g_p(\pi|\lambda) = d_1 + \alpha d_2 \\ \text{where} \quad & d_1 = \frac{\|(F(\pi) - z^*)^\top \lambda\|}{\|\lambda\|} \\ & d_2 = \|F(\pi) - (z^* + d_1 \lambda)\|, \pi \in \mathcal{X} \end{aligned} \tag{4}$$

where $\alpha > 0$ is a preset penalty item and z^* is the ideal reference point as defined in the Tchebycheff approach. The PBI approach can yield more uniformly distributed PFs than the Tchebycheff approach, especially in the case of more than two objectives [52].

3.2.2 SOVRP Solving. After the decomposition, the subproblems (i.e., SOVRP) are solved using evolutionary algorithms. We outline the typical MOEA/D procedure with the Tchebycheff decomposition in Algorithm 1. In particular, the MOEA/D groups subproblems according to the Euclidean distances of their associated preferences. Initially, a population of N solutions are generated for corresponding subproblems (line 1). Then, the MOEA/D iteratively evolves the population. More specifically, given a target subproblem and its group, another two subproblems in the same group are randomly selected with their solutions used to reproduce an offspring solution by genetic operations (lines 4-5). The offspring would replace the best-so-far solution to each subproblem in the group (including the target one), if it was more elite for each SOVRP (line 6).

However, evolutionary algorithms often deliver inferior solutions for MOVRPs in practice, which heavily depend on considerable iterations to evolve the population and massive labor to tune algorithmic components. In this paper, we parameterize DRL agents by the PAN to automatically learn policies that can efficiently construct solutions to SOVRPs in parallel. We also propose the CAS to further improve the performance, in which we realize effective gradient-based interactions between agents by imitation learning.

4 METHODOLOGY

Given a set of subproblems (i.e., SOVRPs) that are decomposed from an MOVRP, we aim to train the policies for constructing solutions to SOVRPs in parallel. In doing so, we can efficiently solve a set of SOVRPs with respective preferences, yielding the approximate Pareto solutions. To this end, we first model the process of constructing the solution to each SOVRP as a Markov decision process (MDP). Then, we parameterize the policies by the preference-based attention network (PAN), which are trained by DRL algorithm to solve SOVRPs with respective preferences in sync. During inference, we further propose the collaborative active search (CAS) to improve the quality of Pareto solutions per instance, which adopts the PAN to iteratively sample solutions to subproblems and fine-tunes the policies of DRL agents in a collaborative manner. We detail our collaborative DRL method in the following sections.

4.1 DRL for Solving Subproblem

Given the SOVRP with a preference λ_i, we construct the solution following Markov Decision Process (MDP). An *agent* iteratively takes the current *state* as input (e.g., the instance information and the partially constructed solution), and outputs the probabilities of nodes to be visited next. The *action* is a node that is greedily selected or randomly sampled from the probabilities. The *transition* dynamics is joining the node to the partial solution. We parameterize the *policy p* of the agent by a neural network p_θ, so that the probability of constructing a complete tour π to the SOVRP is expressed by $p_\theta(\pi|\mathcal{G}) = \prod_{t=1}^{T} p_\theta(\pi_t|\pi_{<t}, \mathcal{G})$, where π_t and $\pi_{<t}$ represent the selected node and partial solution at the t-th step. Typically, the *reward* is defined as the negative value of the scalarized objective, e.g., $\mathcal{R}(\pi) = -g_t(\pi|\lambda_i, z^*)$ with Tchebycheff decomposition. The policy network is commonly trained with REINFORCE [44] algorithm, which maximizes $\mathcal{L}(\theta|\mathcal{G}) = \mathbb{E}_{p_\theta(\pi|\mathcal{G})}\mathcal{R}(\pi)$ by the gradient,

$$\nabla_\theta \mathcal{L}(\theta|\mathcal{G}) = \mathbb{E}_{p_\theta(\pi|\mathcal{G})}[(\mathcal{R}(\pi) - b(\mathcal{G}))\nabla_\theta \log p_\theta(\pi|\mathcal{G})], \tag{5}$$

where the baseline $b(\cdot)$ reduces the gradient variance and stabilizes the training over solutions to different instances, i.e., $\pi \sim p_\theta, \mathcal{G} \in \tilde{\mathcal{G}}$. There are two paradigms to extend the above DRL to solve MOVRPs. On the one hand, the methods in one-to-one paradigm sequentially train individual neural networks to solve SOVRPs with a predefined set of preferences [25, 54]. However, they generally suffer from low training efficiency and perform inferior on the SOVRPs with preferences unseen during training. On the other hand, the methods in one-to-many paradigm train a single neural network to solve SOVRP with any preference, which incorporates a sub-network to transform preferences into parameters in the neural network, inducing policies tailored for each SOVRP. Following the latter, we propose the PAN to parameterize the policies for SOVRPs. In contrast to existing methods, we design a hybrid intervention strategy to generate both the preference-based and preference-agnostic parameters for gaining more effective policies.

4.2 Preference-based Attention Network

The PAN consists of an encoder shared by SOVRPs and a decoder tailored for each SOVRP by the hybrid intervention. The shared encoder stacks multiple multi-head self-attention layers, which processes each MOVRP instance (in parallel) to attain d_h-dimensional node embeddings $\omega = \{h_j\}_{j=1}^{n}, j \in \mathcal{V}$. Following existing methods, we directly exploit the encoder in Attention Model [21], which has shown decent performance in solving single-objective VRPs.

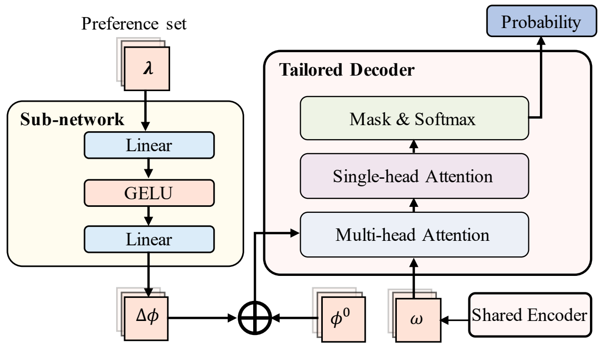

Figure 1: Illustration of the PAN with hybrid intervention.

We attain ω via a single forward pass through the encoder, and then employ the decoder to construct solutions to SOVRPs in an autoregressive manner. Taking MOTSP as an example, the decoder obtains a context embedding $h_c = [h_{\pi_1}; h_{\pi_{t-1}}]$ at each step t, where $h_{\pi_1}, h_{\pi_{t-1}}$ are the first and last selected node in the current partial solution, and $[;]$ means the concatenation operation. Then, we process the context embedding h_c and node embeddings ω by a multi-head attention (MHA) layer as follows,

$$q_c^m = W_Q^m(\lambda_i)h_c;\ k_j^m = W_K^m(\lambda_i)h_j;\ v_j^m = W_V^m(\lambda_i)h_j, \tag{6}$$

$$\delta_{cj}^m = \begin{cases} (q_c^m)^\top k_j^m/\sqrt{d_k}, & \text{if node } j \text{ is valid,} \\ -\infty, & \text{otherwise} \end{cases} \tag{7}$$

$$a_c^m = \sum\nolimits_j u_{cj}^m v_j^m,\ u_{cj}^m = e^{\delta_{cj}^m}/\sum\nolimits_{j'} e^{\delta_{cj'}^m}\ \ m = 1, \cdots, M, \tag{8}$$

$$f_c = W_O(\lambda_i)[a_c^1; \cdots; a_c^M], \tag{9}$$

where Eqs. (6), (7), and (8) formulate the attention computations in M heads; $W_Q^m(\lambda_i) \in R^{d_k \times (2*d_h)}$, $W_K^m(\lambda_i)$, $W_V^m(\lambda_i)$, and $W_O(\lambda_i) \in R^{d_k \times d_h}$ with $d_k = d_h/M$ are trainable parameters, which are derived from the preference λ_i. Subsequently, we obtain the probabilities over all valid nodes via a single-head attention layer as follows,

$$\zeta_j = \begin{cases} C \cdot \tanh({f_c}^\top h_j/\sqrt{d_k}), & \text{if node } j \text{ is valid,} \\ -\infty, & \text{otherwise} \end{cases} \tag{10}$$

$$p_\theta(\pi_t = j|\pi_{<t}, \mathcal{G}, \lambda_i) = e^{\zeta_j}/\sum\nolimits_{j'} e^{\zeta_{j'}}, \tag{11}$$

where $C = 10$ is the clipping value used for better exploration, and the invalid nodes are masked by the softmax operation in Eq. (11).
Hybrid Intervention. To tailor the decoder for each SOVRP, we generate parameters $\phi_i = [W_Q^m(\lambda_i), W_K^m(\lambda_i), W_V^m(\lambda_i), W_O(\lambda_i)]$ in the MHA layer based on respective preferences $\boldsymbol{\lambda} = \{\lambda_i\}_{i=1}^{N}$. However, directly applying a sub-network to transform preferences into the parameters may introduce a proportionality dependency of the parameters on preferences, thereby limiting the capacity of the decoder to learn more effective policies [34].

To mitigate this issue, we propose to employ a sub-network to yield the additive adjustments to the parameters $\phi = \{\phi_i\}_{i=1}^{N}$, rather than ϕ themselves. As illustrated in Figure 1, we initialize a set of trainable parameters $\phi^0 = [W_{Q_0}^m, W_{K_0}^m, W_{V_0}^m, W_{O_0}]$, which are independent of preferences. Then, we apply a multi-layer perception (MLP) as the sub-network to map the preference set $\boldsymbol{\lambda}$ to the additive adjustments $\triangle\phi$, i.e., $\triangle\phi = MLP(\boldsymbol{\lambda})$ with $\boldsymbol{\lambda} \in R^{N\times\kappa}$ and

Algorithm 2 Collaborative Active Search

```
Require: The PAN $p_{\mathcal{H}}$, number of steps $E$, number of subproblems $N$, the neighborhood size $N_s$, number of tours $K$.
1: Initialize a set of $N$ uniformly distributed preferences $\boldsymbol{\lambda} = \{\lambda_1, \cdots, \lambda_N\}$ and their neighborhoods $\{\psi^i, \cdots, \psi^N\}$.
2: Initialize the best-so-far solutions to each subproblem, i.e., $\tilde{\pi}^i = \pi_0^i \sim \text{SampleTour}(p_{\mathcal{H}}(\cdot|\mathcal{G}_s, \lambda_i)), \forall i \in \{1, \cdots, N\}$
3: for $e = 1$ to $E$ do
4:     Record log-probability of generating $\tilde{\pi}^i$ in each step.
5:     Calculate gradient $\nabla_{\mathcal{H}}\mathcal{L}_{CAS}$ with $\tilde{\pi}^i$, $\pi_{e-1}^i$ by Eq. (15)
6:     $\mathcal{H} \leftarrow \textbf{ADAM}(\mathcal{H}, \nabla_{\mathcal{H}}\mathcal{L}_{CAS})$
7:     $\pi_e^i \sim \text{SampleTour}(p_{\mathcal{H}}(\cdot|\mathcal{G}_s, \lambda_i)), \forall i \in \{1, \cdots, N\}$
8:     Evaluate solutions to $\psi^i$ on the $i$th subproblem in parallel, and attain the most elite solution $\pi_*^i, \forall i \in \{1, \cdots, N\}$
9:     Update the best-so-far solution $\tilde{\pi}^i$ to the $i$th subproblem with $\pi_*^i$ if $g_t(\pi_*^i|\lambda_i) < g_t(\tilde{\pi}^i|\lambda_i), \forall i \in \{1, \cdots, N\}$
10: end for
11: return $\{\tilde{\pi}^i\}_{i=1}^N$.
```

$\triangle\phi \in R^{N\times(5*d_h*d_h)}$. After that, we reshape $\triangle\phi$ into $\{\triangle\phi_i\}_{i=1}^N$ for SOVRPs, where $\triangle\phi_i = [\triangle W_Q^m(\lambda_i), \triangle W_K^m(\lambda_i), \triangle W_V^m(\lambda_i), \triangle W_O(\lambda_i)]$ shares the same dimensions as ϕ_i. Accordingly, the final parameters ϕ used in the MHA layer of the decoder is the sum of initial parameters and additive adjustments, i.e., $\phi = \{\phi^0 + \triangle\phi_i\}_{i=1}^N$. Notably, we generate the parameters for SOVRPs with respective N preferences in parallel and ensure that we gain different parameters to intervene policies of each DRL agent.

Policy Optimization. In each batch during training, we distribute the parameters derived from different preferences to random MOVRP instances rather than the same one, enhancing the exploration over both preference and problem spaces. In specific, given a batch of N instance-preference pairs $\{(\mathcal{G}_i, \lambda_i)\}_{i=1}^N$, we sample tours (i.e., solutions) $\{\pi^i\}_{i=1}^N$ for the SOVRPs with respective preferences, and update the PAN with the estimated gradient as below,

$$\nabla_{\theta,\phi}\mathcal{L}(\theta, \phi|\mathcal{G}_i, \lambda_i) \simeq \frac{1}{N}\sum_{i=1}^N (\mathcal{R}(\pi^i|\mathcal{G}_i, \lambda_i) - b(\mathcal{G}_i, \lambda_i))\nabla_{\theta,\phi}\log p_{\theta,\phi}(\pi^i|\mathcal{G}_i, \lambda_i), \tag{12}$$

where we use the expected reward as the baseline $b(\cdot)$ over random training samples, i.e., $\pi^i \sim p_{\theta,\phi}$, $\mathcal{G}_i \in \tilde{\mathcal{G}}$, $\lambda_i \in \tilde{\lambda}$, according to [21].

4.3 Collaborative Active Search

After training, the proposed PAN is able to address multiple subproblems in sync without the need of additional search procedures. While our experiments show it outperforms previous DRL methods, we further enhance its performance from two considerations. First, existing DRL methods for MOVRPs do not sufficiently leverage the collaboration between agents, especially those addressing similar subproblems. Intuitively, these agents could share valuable knowledge and assist each other in finding better solutions to respective subproblems (i.e., Pareto solutions). Second, deep models often degenerate when tested on configurations (e.g., problem sizes or node distributions) beyond the training ones, limiting their applications in diverse scenarios. To mitigate the above issues, we propose the CAS to search better Pareto solutions per instance.

Given a set of subproblems with $\{\lambda_1, \cdots, \lambda_N\}$, we define a neighborhood ψ_i of a subproblem with λ_i as a subset of subproblems with closest preferences, which are measured by Euclidean distances between preferences. Then, we fine-tune the PAN on an MOVRP instance, by iteratively sampling solutions to SOVRPs and updating (instance-specific) policies of DRL agents. Imitation learning is used within each neighborhood to further enhance the performance.

Self-Active Search. The active search is proposed in [2] and firstly used for MOVRPs in [28]. As a simple yet effective gradient-based search approach, it is focused on adapting a pre-trained model on a single problem instance. While improving the solution quality, the vanilla active search is prohibitively time-consuming to train the whole neural network for each test instance. Instead, we only update a limited number of parameters in the PAN while keeping the others fixed. In doing so, much less gradient-related computation is needed in comparison to vanilla active search, which could significantly increase the efficiency of the search process.

Specifically, we initialize a trainable matrix $\mathcal{H} \in R^{n\times d_h}$ with the values in node embeddings $\omega' = \{h_j'\}_{j=1}^n$, $j \in \mathcal{V}$, which are derived from the trained encoder in the PAN. During the active search for a specific instance $\mathcal{G}_s$, we substitute the above matrix to $\{h_j\}_{j=1}^n$ in Eq. (10) to gain the probabilities over nodes, and update it following the similar estimated gradient displayed in Eq. (12), such that,

$$\nabla_{\mathcal{H}}\mathcal{L}(\mathcal{H}|\mathcal{G}_s, \lambda_i) \simeq \frac{1}{N}\sum_{i=1}^N (\mathcal{R}(\pi^i|\mathcal{G}_s, \lambda_i) - b(\mathcal{G}_s, \lambda_i))\nabla_{\mathcal{H}}\log p_{\mathcal{H}}(\pi^i|\mathcal{G}_s, \lambda_i), \tag{13}$$

where $p_{\mathcal{H}}$ means the PAN with only $\mathcal{H}$ to be trained; π^i means the sampled solution to the ith subproblem with preference λ_i in the instance $\mathcal{G}_s$, which satisfies $p_{\mathcal{H}}(\pi^i|\mathcal{G}_s, \lambda_i) = \Pi_{t=1}^T p_{\mathcal{H}}(\pi_t^i|\pi_{<t}^i, \mathcal{G}_s, \lambda_i)$. $b(\cdot)$ is a baseline with the same definition as in Eq. (12). Please note that we prevent gradients from flowing backward through the encoder, and thus avoid considerable computation. Moreover, the above update is used to fine-tune a small part of PAN, which is shared by each DRL agent. Therefore the policies of agents to solve corresponding SOVRPs are swiftly adapted.

Interaction by Imitation Learning. In addition to the above DRL, where agents update their policies with respective rewards ($\mathcal{R}(\pi^i|\mathcal{G}_s, \lambda_i)$), we facilitate interactions among the agents to further enhance their collaborative performance. Without loss of generality, we take the neighborhood ψ^i of the subproblem with λ_i as an example. Specifically, we initialize the best-so-far solution $\tilde{\pi}^i$ to the ith subproblem with a solution π_0^i sampled by the trained PAN. At each active search step, we foster the agent for the ith subproblem to generate the solution $\tilde{\pi}^i$ by imitation learning, thus adjusting the output probabilities of PAN towards generating $\tilde{\pi}^i$. Accordingly, we maximize $\mathcal{L}(\mathcal{H}|\mathcal{G}_s) = \mathbb{E}_{\lambda_i} p_{\mathcal{H}}(\tilde{\pi}^i|\mathcal{G}_s, \lambda_i)$ in the CAS, by updating PAN with the log-probability of generating $\tilde{\pi}^i$ in each step, as below,

$$\begin{aligned}\nabla_{\mathcal{H}}\mathcal{L}_I(\mathcal{H}|\mathcal{G}_s, \lambda_i) &\simeq \tfrac{1}{N}\textstyle\sum_{i=1}^N \nabla_{\mathcal{H}}\log p_{\mathcal{H}}(\tilde{\pi}^i|\mathcal{G}_s, \lambda_i) \\ &= \tfrac{1}{N}\textstyle\sum_{i=1}^N\sum_{t=1}^T \nabla_{\mathcal{H}}\log p_{\mathcal{H}}(\tilde{\pi}_t^i|\tilde{\pi}_{<t}^i, \mathcal{G}_s, \lambda_i)\end{aligned} \tag{14}$$

After update in each step, we sample solutions to all subproblems, evaluate the solutions to ψ^i in parallel on the ith subproblem, and attain the most elite solution π_*^i. We replace the best-so-far solution $\tilde{\pi}^i$ with π_*^i if the latter is better. We repeat the above process in each step until the termination. The rationale behind the interaction is that the elite solutions to respective subproblems could also be favorable and informative to their similar neighboring subproblems.

Consequently, the overall gradient used to update the PAN in the CAS is defined as below,

$$\nabla_{\mathcal{H}}\mathcal{L}_{CAS} = \nabla_{\mathcal{H}}\mathcal{L}(\mathcal{H}|\mathcal{G}_s, \lambda_i) + \sigma\nabla_{\mathcal{H}}\mathcal{L}_I(\mathcal{H}|\mathcal{G}_s, \lambda_i), \tag{15}$$

where σ is a hyperparameter used to balance the exploration by $\mathcal{L}$ and exploitation by $\mathcal{L}_I$. A larger σ could encourage agents to be more focused on generating the historical elite solutions in the neighborhood. We describe the CAS by pseudocode in Algorithm 2.

5 EXPERIMENTS

5.1 Experimental Setting

Problems&Training. We conduct extensive experiments to evaluate the effectiveness of the proposed PAN and CAS across various MOVRPs, i.e., bi-objective TSP (BiTSP), tri-objective TSP (TriTSP), and bi-objective CVRP (BiCVRP), as did in the existing works [28, 30, 54]. Regarding the κ-objective TSP, each node i is featured by κ 2D-coordinates, and the κ-th cost between node i and j is calculated as the Euclidean distance between their κ-th 2D-coordinates. Regarding BiCVRP, the conflicting objectives are the minimization of the total tour length and the length of the longest route, respectively, as per prior works [17, 22]. Following [28], we use Tchebycheff and PBI approaches to scalarize bi-objective VRPs (i.e., BiTSP and BiVRP) and tri-objective VRPs (i.e., TriTSP), respectively. During training, we utilize consistent hyperparameters across all problems throughout our experiments. Specifically, we train the PAN for each problem with the problem size defined by the number of customer nodes (i.e., 50 and 100). We generate 100,000 instances on the fly in each epoch. The 2D coordinates and demands (in BiCVRP instances) are uniformly sampled from the range $[0, 1]^2$ and the discrete set $\{1, \ldots, 9\}$, respectively. We set the batch size to 64, and train the PAN with 200 epochs using the Adam optimizer. The learning rate is set to 10^{-4} with a decay rate 10^{-6}.

Baselines. We compare our trained PAN models with typical established optimization algorithms for MOVRPs, encompassing both conventional MOEAs and recent deep models for MOVRPs. Specifically, the baselines in the comparison include: 1) **MOEA/D** [52], a classic decomposition-based MOEA executed with 4,000 iterations; 2) **NSGA-II** [9], a representative Pareto dominance-based multi-objective genetic algorithm implemented with 4,000 iterations; 3) **MOGLS** [16], a multi-objective genetic local search algorithm executed with 10,000 iterations and 100 local search steps in each iteration; 4) **NSGA-III** [4], an extension of NSGA-II by introducing reference direction, which is implemented with 4,000 iterations; 5) **DRL-MOA** [25], a DRL approach that decomposes an MOVRP into SOVRPs under different preferences and trains individual solving policies with the parameter transfer scheme; 6) **ML-DAM** [54], an approach that fine-tunes a meta-model to obtain individual models for solving respective SOVRPs; 7) **PMOCO** [28], a deep model that solves SOVRPs with the preference-based hypernetwork, which achieves state-of-the-art performance in solving MOTSP and MOCVRP among existing deep models. We train DRL-MOA and PMOCO following their original training settings. For ML-DAM, we set the number of iterations to 5,000 for BiCVRP50 and 1,000 for BiCVRP100, ensuring similar training overhead among deep models. The rest is the same as those in the original paper [54]. Regarding MOEAs, we apply 2-opt and problem-specific local operators for solving MOTSPs and BiCVRP, respectively, following the work [22].

Hyperparameters of Active Search. We use the Adam optimizer for both our CAS and the vanilla active search (used in PMOCO). Regarding CAS, we set the learning rate to 0.0041 with a decay rate of 10^{-6}, and set the indicator σ to 0.013 for balancing exploration and exploitation. Regarding the vanilla active search, we choose a learning rate of 2.6e-5 with a decay rate of 10^{-6} for its best performance. For the termination condition, we set 200 iterations for both CAS and the vanilla active search, according to [15].

Metrics&Inference. As mentioned above, the trained PAN can be evaluated in two modes, i.e., the direct test and CAS. Correspondingly, PMOCO can also be directly tested and equipped with the vanilla active search as used in its original paper [28]. For the direct test, we implement two versions for PAN and PMOCO, i.e., with and without instance augmentation. The instance augmentation is often used in the literature to generate multiple solutions to symmetric transformations of an instance. To ensure reasonable inference time, we use 8 transformations for the direct test on BiVRP and 64 transformations for the direct test on BiTSP and TriTSP in PAN and PMOCO, respectively. Regarding PAN with CAS and PMOCO with vanilla active search, we use 8 transformations to solve all problems. We assess all methods using three metrics: average hypervolume (HV) [43], average gap, and total runtime per instance set. Among them, HV serves as a widely used metric in multi-objective optimization to reflect both the convergence and diversity of Pareto solutions. To keep uniformity in experiments, we normalize HV values to the range of $[0, 1]$ using the same reference point for all methods, so that higher HV values indicate better performance. The gap is defined as the ratio of the hypervolume difference relative to the best HV among all methods. We denote the best average HV and gap among all methods by **bold** throughout the paper. PAN and baseline methods are implemented in Python on a device with RTX A100 GPU and Intel Xeon Gold 6226R CPU.

5.2 Comparison Study

We compare our PAN and CAS with the baselines on BiTSP, BiCVRP, and TriTSP across four sizes (i.e., 50, 100, 150, and 200). Deep models (i.e., DRL-MOA, ML-DAM, PMOCO, and PAN) trained on size 50 are evaluated on identical-sized instances, while those trained on size 100 are tested across sizes 100 to 200. Notably, DRL-MOA and ML-DAM train individual models for bi/tri-objective VRPs with 101 and 105 uniformly distributed preferences, respectively. For fairness, we apply the same preference sets to other decomposition-based methods (i.e., MOEA/D, PMOCO, and PAN) during the test.

We summarize the results in Table 1, where we append "(Aug=X)" to PMOCO or PAN signifying the usage of instance augmentation with the indicated number of transformations. It is clear that without instance augmentation and active search (AS), the trained PAN outperforms all baselines across most instance sets, except for a slight loss to PMOCO on BiTSP50. Especially on BiCVRP100, BiCVRP150, and BiCVRP200, we observe the PAN significantly surpasses the other DRL methods (i.e., DRL-MOA, ML-DAM, and PMOCO) with over 10% smaller gaps. It also indicates the fairly good zero-shot generalization of PAN on problem sizes 150 and 200. On

Table 1: Results on MOVRPs with different problem sizes. Left: Test on 200 instances. Right: Generalization on 20 instances.

	BiTSP50			BiTSP100			BiTSP150			BiTSP200		
Methods	HV	Gap	Time	HV	Gap	Time	HV	Gap	Time	HV	Gap	Time
MOEA/D	0.552	4.31%	25.83h	0.589	13.28%	39.65h	0.582	19.97%	5.28h	0.574	23.87%	6.71h
NSGA-II	0.561	2.87%	18.11h	0.580	14.58%	32.13h	0.546	24.85%	4.94h	0.509	32.55%	6.27h
MOGLS	0.521	9.71%	13.20h	0.558	17.90%	34.40h	0.564	22.45%	9.79h	0.492	34.72%	14.91h
NSGA-III	0.522	9.62%	18.57h	0.565	16.88%	33.95h	0.536	26.27%	4.94h	0.562	25.48%	6.57h
DRL-MOA	0.491	15.02%	6.03s	0.605	10.95%	12.06s	0.657	9.58%	7.41s	0.686	9.05%	8.51s
ML-DAM	0.554	3.98%	6.43s	0.638	6.14%	9.93s	0.684	5.83%	10.48s	0.712	5.57%	15.27s
PMOCO	0.567	1.84%	12.67s	0.664	2.24%	39.79s	0.710	2.34%	15.90s	0.736	2.51%	29.94s
PMOCO(Aug=64)	0.573	0.81%	8.60m	0.670	1.38%	38.42m	0.714	1.69%	12.80m	0.740	1.92%	26.31m
PMOCO(AS, Aug=8)	0.576	0.19%	5.93h	0.677	0.41%	23.37h	0.723	0.45%	30.51h	0.752	0.34%	39.89h
PAN	0.566	1.89%	12.22s	0.666	1.94%	43.31s	0.713	1.90%	16.22s	0.740	1.92%	28.80s
PAN(Aug=64)	0.574	0.64%	9.14m	0.672	1.09%	39.19m	0.718	1.17%	12.11m	0.745	1.30%	24.99m
PAN(CAS, Aug=8)	**0.577**	**0.00%**	2.20h	**0.680**	**0.00%**	10.69h	**0.727**	**0.00%**	2.76h	**0.754**	**0.00%**	4.44h

	BiCVRP50			BiCVRP100			BiCVRP150			BiCVRP200		
Methods	HV	Gap	Time	HV	Gap	Time	HV	Gap	Time	HV	Gap	Time
MOEA/D	0.277	1.74%	44.26h	0.253	6.41%	75.26h	0.242	18.10%	10.77h	0.183	35.48%	13.92h
NSGA-II	0.278	1.47%	38.07h	0.237	12.30%	70.29h	0.222	24.79%	10.86h	0.183	35.38%	14.22h
MOGLS	0.272	3.38%	50.15h	0.240	10.97%	92.74h	0.243	17.45%	13.46h	0.229	19.08%	19.48h
NSGA-III	0.278	1.54%	36.48h	0.238	11.92%	73.96h	0.227	22.95%	10.95h	0.180	36.41%	14.43h
DRL-MOA	0.247	12.34%	10.26s	0.238	11.82%	25.55s	0.263	10.85%	20.24s	0.253	10.81%	28.92s
ML-DAM	0.237	15.82%	6.79s	0.207	23.42%	12.29s	0.230	22.14%	13.26s	0.232	18.11%	16.57s
PMOCO	0.278	1.56%	13.59s	0.221	18.16%	48.56s	0.235	20.48%	19.74s	0.218	23.13%	36.36s
PMOCO(Aug=8)	0.280	0.71%	1.37m	0.235	12.94%	5.95m	0.250	15.09%	1.93s	0.235	16.95%	3.95m
PMOCO(AS, Aug=8)	0.282	0.14%	6.93h	0.269	0.30%	23.08h	0.293	0.71%	38.74h	0.283	0.11%	50.13h
PAN	0.278	1.31%	14.25s	0.266	1.41%	51.32s	0.288	2.51%	19.80s	0.275	3.04%	35.65s
PAN(Aug=8)	0.280	0.60%	1.43m	0.268	0.74%	6.20m	0.290	1.63%	1.92m	0.275	2.93%	4.25m
PAN(CAS, Aug=8)	**0.282**	**0.00%**	3.37h	**0.270**	**0.00%**	11.35h	**0.295**	**0.00%**	5.28h	**0.283**	**0.00%**	8.37h

	TriTSP50			TriTSP100			TriTSP150			TriTSP200		
Methods	HV	Gap	Time	HV	Gap	Time	HV	Gap	Time	HV	Gap	Time
MOEA/D	0.184	48.30%	29.28h	0.230	49.77%	43.33h	0.252	50.63%	5.60h	0.264	51.34%	7.04h
NSGA-II	0.173	51.36%	18.29h	0.155	66.11%	35.06h	0.134	73.74%	5.00h	0.114	78.94%	6.42h
MOGLS	0.292	17.77%	29.45h	0.324	29.42%	75.09h	0.327	35.93%	14.02h	0.325	40.08%	21.45h
NSGA-III	0.255	28.27%	20.73h	0.258	43.76%	34.90h	0.234	54.13%	5.25h	0.207	61.79%	7.11h
DRL-MOA	0.237	33.40%	8.71s	0.354	22.80%	14.25s	0.407	20.40%	8.45s	0.442	18.59%	10.57s
ML-DAM	0.284	20.20%	5.98s	0.337	26.47%	10.84s	0.383	25.02%	13.70s	0.414	23.79%	17.74s
PMOCO	0.341	4.19%	11.82s	0.439	4.25%	41.41s	0.491	3.97%	16.83s	0.523	3.67%	29.97s
PMOCO(Aug=64)	0.349	1.91%	9.48m	0.447	2.55%	39.75m	0.497	2.68%	13.15m	0.529	2.60%	26.46m
PMOCO(AS, Aug=8)	0.352	0.93%	6.54h	0.455	0.92%	23.88h	0.507	0.72%	31.74h	**0.543**	**0.00%**	41.38h
PAN	0.342	3.74%	12.28s	0.442	3.66%	42.09s	0.493	3.50%	16.92s	0.524	3.39%	29.83s
PAN(Aug=64)	0.350	1.55%	9.47m	0.449	2.11%	40.69m	0.500	2.21%	12.58m	0.530	2.30%	25.93m
PAN(CAS, Aug=8)	**0.355**	**0.00%**	2.18h	**0.459**	**0.00%**	8.53h	**0.511**	**0.00%**	2.71h	0.542	0.15%	4.61h

the other hand, the performance of PAN is further elevated by the use of instance augmentation and CAS. In particular, PAN(Aug=64) is superior to PMOCO(Aug=64) with similar runtime across all instance sets. PAN(CAS, Aug=8) attains the best results across all instance sets except for TriTSP200, where it is narrowly edged out by PMOCO(AS, Aug=8). In terms of computational efficiency, conventional heuristics consume significantly more runtime than deep models, with their inferiority turning more apparent as the problem size grows. PAN and PMOCO take comparable runtime, which is a little longer than that of DRL-MOA and ML-DAM. On the other hand, the proposed CAS consumes extra inference time on top of the PAN to deliver much better solutions. Notably, the PAN with the CAS remains more efficient than PMOCO with the vanilla active search, as well as conventional heuristics.

5.3 Benchmark Study

To further verify the effectiveness of the PAN and CAS, we compare them with the baselines on 8 benchmark instances for MOTSP, including 7 instances of BiTSP100 (i.e., KroAB100, KroAC100, KroAD100, KroBC100, KroBD100, KroCD100, and ClusAB100) and 1 instance of BiTSP150 (i.e., KroAB150). These instances are generated according to [29] and commonly used in the literature [25, 28, 54]. All results are shown in Table 2, where we report the results of exact Pareto fronts that are obtained by prohibitively exhaustive search in [12]. We mark the best results except the exact solutions in bold. As shown, when the instance augmentation is not used, PAN achieves significantly smaller HV and gap than the other baselines. PAN(Aug=64) with instance augmentation further promotes the performance of PAN and outperforms PMOCO(Aug=64). In addition, PAN(CAS, Aug=8) outperforms PMOCO(AS, Aug=8) across all instances, with less than 0.5% and 1% optimality gap for BiTSP100 and BiTSP150, respectively. Notably, the runtime of PAN(CAS, Aug=8) is about 20 times less than that of PMOCO(AS, Aug=8) across all instances. Since the node distributions in the benchmark instances differ considerably from the distribution in training, the results manifest that the PAN has a strong out-of-distribution generalization capability and the CAS can further enhance the performance remarkably.

5.4 Ablation Study

Preference-agnostic Parameters. We verify the effect of the proposed preference-agnostic parameters in the hybrid intervention. In specific, we remove these parameters from PAN, resulting in a model termed PAN w/o PAP. We compare it with the original

Table 2: Results on benchmark instances.

Method	KroAB100 HV	KroAB100 Gap	KroAB100 Time	KroAC100 HV	KroAC100 Gap	KroAC100 Time	KroAD100 HV	KroAD100 Gap	KroAD100 Time	KroBC100 HV	KroBC100 Gap	KroBC100 Time
Exact Pareto front	0.782	0.00%	58h	0.783	0.00%	30h	0.785	0.00%	21h	0.784	0.00%	28h
MOEA/D	0.703	10.10%	7.72m	0.693	11.57%	7.72m	0.701	10.76 %	7.72m	0.699	10.77%	7.72m
NSGA-II	0.696	10.98%	6.52m	0.680	13.15%	6.52m	0.692	11.80%	6.52m	0.707	9.85%	6.52m
MOGLS	0.684	12.48%	10.50m	0.687	12.24%	10.50m	0.686	12.66%	10.50m	0.687	12.36%	10.58m
NSGA-III	0.681	12.89%	9.17m	0.655	16.37%	9.08m	0.681	13.27%	9.08m	0.675	13.82%	9.10m
DRL-MOA	0.560	28.34%	3.95s	0.585	25.32%	3.62s	0.586	25.38%	3.63s	0.612	21.88%	3.6s
ML-DAM	0.739	5.49%	6.52s	0.746	4.81%	6.23s	0.734	6.54%	6.17s	0.733	6.47%	6.28s
PMOCO	0.759	2.84%	3.84s	0.760	2.92%	3.43s	0.764	2.65%	3.51s	0.762	2.83%	3.62s
PMOCO(Aug=64)	0.767	1.89%	13.45s	0.768	1.99%	13.08s	0.771	1.73%	13.08s	0.768	1.99%	13.08s
PMOCO(AS, Aug=8)	0.776	0.72%	38.21m	0.777	0.75%	36.92m	0.780	0.70%	36.95m	0.778	0.69%	38.65m
PAN	0.766	1.97%	0.58s	0.766	2.20%	0.29s	0.768	2.14%	0.24s	0.766	2.33%	0.20s
PAN(Aug=64)	0.771	1.37%	12.22s	0.772	1.47%	11.88s	0.774	1.39%	11.91s	0.773	1.44%	11.9s
PAN(CAS, Aug=8)	**0.778**	**0.40%**	1.8m	**0.780**	**0.38%**	1.78m	**0.782**	**0.34%**	1.78m	**0.781**	**0.38%**	1.78m

Method	KroBD100 HV	KroBD100 Gap	KroBD100 Time	KroCD100 HV	KroCD100 Gap	KroCD100 Time	ClusAB100 HV	ClusAB100 Gap	ClusAB100 Time	kroAB150 HV	kroAB150 Gap	kroAB150 Time
Exact Pareto front	0.781	0.00%	23h	0.791	0.00%	21h	0.78	0.00%	27h	0.728	0.00%	days
MOEA/D	0.703	9.98%	7.72m	0.704	11.02%	7.72m	0.707	9.30%	10.78m	0.589	19.12%	10.25m
NSGA-II	0.696	10.88%	6.52m	0.704	11.05%	6.52m	0.689	11.68%	8.98m	0.532	26.95%	9.32m
MOGLS	0.689	11.84%	10.40m	0.693	12.42%	10.27m	0.689	11.62%	11.02m	0.549	24.61%	18.03m
NSGA-III	0.667	14.61%	9.08m	0.689	12.89%	9.07m	0.688	11.74%	9.18m	0.536	26.36%	13.35m
DRL-MOA	0.605	22.61%	3.6s	0.586	25.90%	3.6s	0.710	8.99%	3.96s	0.651	10.49%	5.69s
ML-DAM	0.742	5.03%	6.28s	0.751	5.07%	6.27s	0.740	5.14%	7.28s	0.685	5.94%	9.71s
PMOCO	0.760	2.66%	3.42s	0.768	2.94%	3.53s	0.755	3.19%	3.87s	0.703	3.35%	5.64s
PMOCO(Aug=64)	0.767	1.86%	13.07s	0.777	1.86%	13.07s	0.763	2.17%	14.19s	0.709	2.56%	39.52s
PMOCO(AS, Aug=8)	0.775	0.74%	37.89m	0.786	0.67%	37.98m	0.773	0.95%	37.55m	0.719	1.24%	1.06h
PAN	0.764	2.19%	0.21s	0.774	2.17%	0.20s	0.760	2.58%	0.63s	0.708	2.78%	0.97s
PAN(Aug=64)	0.771	1.36%	11.98s	0.780	1.47%	12.11s	0.766	1.77%	12.12s	0.712	2.12%	36.82s
PAN(CAS, Aug=8)	**0.779**	**0.32%**	1.78m	**0.788**	**0.37%**	1.78m	**0.777**	**0.44%**	1.79m	**0.722**	**0.81%**	8.24m

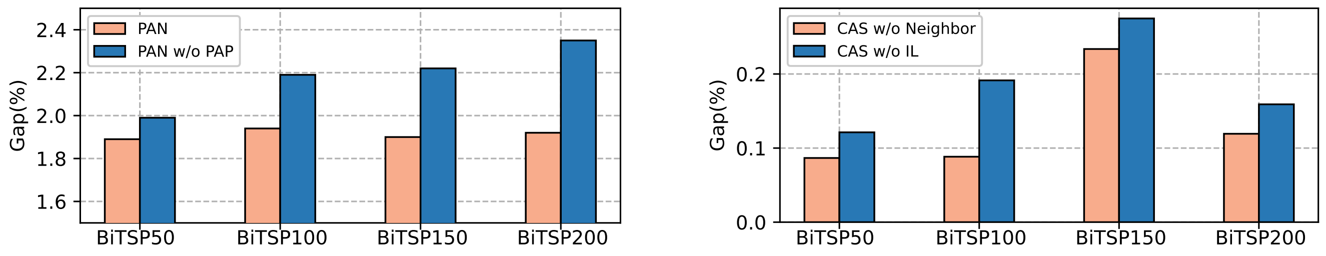

Figure 2: Ablation study. Left: Effect of preference-agnostic parameters. Right: Effects of neighborhood and imitation learning.

PAN on BiTSP50, BiTSP100, BiTSP150, and BiTSP200 by recording their gaps with respect to the best results in Table 1. As shown in the left part of Figure 2, the absence of preference-agnostic parameters leads to a consistent degradation in the performance of PAN, which is particularly evident as the problem size increases. Therefore, our design of the preference-agnostic parameters shows more effectiveness upon the preference-based intervention.

Components of CAS. We further explore the impact of important components in the CAS, including the neighborhood definition and the interaction by imitation learning. We ablate their effects on the performance by gradually removing them from CAS. We first remove the usage of neighborhood from CAS, resulting in the variant denoted by CAS w/o Neighbor, and then remove the imitation learning from CAS w/o Neighbor, resulting in the variant denoted by CAS w/o IL. We compare them with the original CAS on BiTSP50, BiTSP100, BiTSP150, and BiTSP200, and visualize their HV gaps relative to the original CAS in the right part of Figure 2. We observe that the performance of the original CAS significantly degrades when any of the components is removed, which reveals the effectiveness of these key designs in our CAS.

6 CONCLUSION

In this paper, we propose a novel collaborative deep reinforcement learning method for solving MOVRPs. We first decompose an MOVRP into a set of SOVRPs with respective preferences and develop the PAN to learn policies for solving the SOVRPs in parallel. Especially, we design a hybrid intervention in the decoder of PAN to tailor policies for the DRL agents. Then, we propose the CAS to further elevate solution quality, which enables agents to exchange insights of elite solutions by imitation learning. Extensive results show the superiority of our method, especially in solving instances beyond training configurations. In the future, we plan to extend our method to solve more general combinatorial optimization problems, e.g., multi-objective job scheduling and network design problems.

ACKNOWLEDGMENTS

This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant. It was also supported by the National Natural Science Foundation of China under Grant 62372081, the Fundamental Research Funds for the Central Universities under grant DUT21TD107, DUT22ZD214.

REFERENCES

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

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

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

[4] Julian Blank, Kalyanmoy Deb, and Proteek Chandan Roy. 2019. Investigating the normalization procedure of NSGA-III. In *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 229–240.

[5] Juan Castro-Gutierrez, Dario Landa-Silva, and José Moreno Pérez. 2011. Nature of real-world multi-objective vehicle routing with evolutionary algorithms. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 257–264.

[6] Xinyun Chen and Yuandong Tian. 2019. Learning to perform local rewriting for combinatorial optimization. In *International Conference on Neural Information Processing Systems*, Vol. 32. 6281–6292.

[7] Eng Ung Choo and Derek R Atkins. 1983. Proper efficiency in nonconvex multicriteria programming. *Mathematics of Operations Research* 8, 3 (1983), 467–470.

[8] Kalyanmoy Deb and Himanshu Jain. 2013. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE transactions on evolutionary computation* 18, 4 (2013), 577–601.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[10] Matthias Ehrgott. 2005. *Multicriteria optimization*. Vol. 491. Springer Science & Business Media.

[11] Wei Fang, Qiang Zhang, Jun Sun, and Xiaojun Wu. 2020. Mining high quality patterns using multi-objective evolutionary algorithm. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3883–3898.

[12] Kostas Florios and George Mavrotas. 2014. Generation of the exact pareto set in multi-objective traveling salesman and set covering problems. *Appl. Math. Comput.* 237 (2014), 1–19.

[13] Carlos García-Martínez, Oscar Cordón, and Francisco Herrera. 2007. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research* 180, 1 (2007), 116–148.

[14] Pascal Halffmann, Luca E Schäfer, Kerstin Dächert, Kathrin Klamroth, and Stefan Ruzika. 2022. Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. *Journal of Multi-Criteria Decision Analysis* 29, 5-6 (2022), 341–363.

[15] André Hottung, Yeong-Dae Kwon, and Kevin Tierney. 2021. Efficient Active Search for Combinatorial Optimization Problems. In *International Conference on Learning Representations*.

[16] Hisao Ishibuchi and Tadahiko Murata. 1998. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28, 3 (1998), 392–403.

[17] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. 2008. Multi-objective vehicle routing problems. *European Journal of Operational Research* 189, 2 (2008), 293–309.

[18] Liangjun Ke, Qingfu Zhang, and Roberto Battiti. 2014. A simple yet efficient multiobjective combinatorial optimization method using decompostion and pareto local search. *IEEE Transactions on Cybernetics* 44 (2014), 1808–1820.

[19] Minsu Kim, Jinkyoo Park, et al. 2021. Learning collaborative policies to solve NP-hard routing problems. In *International Conference on Neural Information Processing Systems*, Vol. 34. 10418–10430.

[20] Wouter Kool, Herke Van Hoof, and Max Welling. 2018. Attention, learn to solve routing problems!. In *International Conference on Learning Representations*.

[21] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. 2020. Pomo: Policy optimization with multiple optima for reinforcement learning. In *International Conference on Neural Information Processing Systems*, Vol. 33. 21188–21198.

[22] Philippe Lacomme, Christian Prins, and Marc Sevaux. 2006. A genetic algorithm for a bi-objective capacitated arc routing problem. *Computers & Operations Research* 33, 12 (2006), 3473–3493.

[23] Jingwen Li, Yining Ma, Zhiguang Cao, Yaoxin Wu, Wen Song, Jie Zhang, and Yeow Meng Chee. 2023. Learning Feature Embedding Refiner for Solving Vehicle Routing Problems. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[24] Ke Li, Alvaro Fialho, Sam Kwong, and Qingfu Zhang. 2013. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 18, 1 (2013), 114–130.

[25] Kaiwen Li, Tao Zhang, and Rui Wang. 2020. Deep reinforcement learning for multiobjective optimization. *IEEE Transactions on Cybernetics* 51, 6 (2020), 3103–3114.

[26] Shicheng Li, Feng Wang, Qi He, and Xujie Wang. 2023. Deep reinforcement learning for multi-objective combinatorial optimization: A case study on multi-objective traveling salesman problem. *Swarm and Evolutionary Computation* (2023), 101398.

[27] Wu Lin, Qiuzhen Lin, Junkai Ji, Zexuan Zhu, Carlos A Coello Coello, and Ka-Chun Wong. 2021. Decomposition-based multiobjective optimization with bicriteria assisted adaptive operator selection. *Swarm and Evolutionary Computation* 60 (2021), 100790.

[28] Xi Lin, Zhiyuan Yang, and Qingfu Zhang. 2022. Pareto Set Learning for Neural Multi-objective Combinatorial Optimization. In *International Conference on Learning Representations*.

[29] Thibaut Lust and Jacques Teghem. 2010. Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics* 16, 3 (2010), 475–510.

[30] Ishaan Mehta, Sharareh Taghipour, and Sajad Saeedi. 2022. Pareto Frontier Approximation Network (PA-Net) to Solve Bi-objective TSP. In *IEEE 18th International Conference on Automation Science and Engineering*. 1198–1205.

[31] Kaisa Miettinen. 2012. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media.

[32] MODHI LAFTA Mutar, A Burhanuddin, SHAKIR HAMEED, N Yusof, MF Alrifaie, and AA Mohammed. 2020. Multi-objectives ant colony system for solving multi-objectives capacitated vehicle routing problem. *Journal of Theoretical and Applied Information Technology* 98, 24 (2020).

[33] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. 2018. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, Vol. 31.

[34] Jose Javier Gonzalez Ortiz, John Guttag, and Adrian Dalca. 2023. Non-Proportional Parametrizations for Stable Hypernetwork Learning. *arXiv preprint arXiv:2304.07645* (2023).

[35] Luís Paquete and Thomas Stützle. 2009. Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Computers & Operations Research* 36, 9 (2009), 2619–2631.

[36] Syed Mohib Raza, Mohammad Sajid, and Jagendra Singh. 2022. Vehicle Routing Problem using Reinforcement Learning: Recent Advancements. In *Advanced Machine Intelligence and Signal Processing*. Springer, 269–280.

[37] Haitham Seada and Kalyanmoy Deb. 2015. A unified evolutionary optimization procedure for single, multiple, and many objectives. *IEEE Transactions on Evolutionary Computation* 20, 3 (2015), 358–369.

[38] Yinan Shao, Jerry Chun-Wei Lin, Gautam Srivastava, Dongdong Guo, Hongchun Zhang, Hu Yi, and Alireza Jolfaei. 2021. Multi-objective neural evolutionary algorithm for combinatorial optimization problems. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[39] Tipwimol Sooktip and Naruemon Wattanapongsakorn. 2015. Identifying preferred solutions for multi-objective optimization: application to capacitated vehicle routing problem. *Cluster Computing* 18 (2015), 1435–1448.

[40] Ye Tian, Xiaopeng Li, Haiping Ma, Xingyi Zhang, Kay Chen Tan, and Yaochu Jin. 2022. Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2022).

[41] Ye Tian, Langchun Si, Xingyi Zhang, Ran Cheng, Cheng He, Kay Chen Tan, and Yaochu Jin. 2021. Evolutionary large-scale multi-objective optimization: A survey. *Comput. Surveys* 54, 8 (2021), 1–34.

[42] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, Vol. 28.

[43] Lyndon While, Philip Hingston, Luigi Barone, and Simon Huband. 2006. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation* 10, 1 (2006), 29–38.

[44] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3 (1992), 229–256.

[45] Hong Wu, Jiahai Wang, and Zizhen Zhang. 2020. MODRL/D-AM: Multiobjective deep reinforcement learning algorithm using decomposition and attention model for multiobjective optimization. In *International Symposium on Intelligence Computation and Applications*. 575–589.

[46] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, Abhishek Gupta, and Mingyan Lin. 2022. Graph Learning Assisted Multi-Objective Integer Programming. *Advances in Neural Information Processing Systems* 35 (2022), 17774–17787.

[47] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. 2021. Learning Improvement Heuristics for Solving Routing Problems. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[48] Yingbo Xie, Shengxiang Yang, Ding Wang, Junfei Qiao, and Baocai Yin. 2022. Dynamic Transfer Reference Point-Oriented MOEA/D Involving Local Objective-Space Knowledge. *IEEE Transactions on Evolutionary Computation* 26, 3 (2022), 542–554.

[49] Te Ye, Zizhen Zhang, Jinbiao Chen, and Jiahai Wang. 2022. Weight-Specific-Decoder Attention Model to Solve Multiobjective Combinatorial Optimization Problems. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2839–2844.

[50] Jiao-Hong Yi, Li-Ning Xing, Gai-Ge Wang, Junyu Dong, Athanasios V Vasilakos, Amir H Alavi, and Ling Wang. 2020. Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Information Sciences* 509 (2020), 470–487.

[51] Sandra Zajac and Sandra Huber. 2021. Objectives and methods in multi-objective routing problems: a survey and classification scheme. *European Journal of Operational Research* 290, 1 (2021), 1–25.

[52] Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (2007), 712–731.

[53] Yongxin Zhang, Jiahai Wang, Zizhen Zhang, and Yalan Zhou. 2021. MODRL/D-EL: Multiobjective Deep Reinforcement Learning with Evolutionary Learning for Multiobjective Optimization. In *International Joint Conference on Neural Networks*. 1–8.

[54] Zizhen Zhang, Zhiyuan Wu, Hang Zhang, and Jiahai Wang. 2022. Meta-Learning-Based Deep Reinforcement Learning for Multiobjective Optimization Problems. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[55] Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. 2023. Towards Omni-generalizable Neural Methods for Vehicle Routing Problems. In *the 40th International Conference on Machine Learning (ICML 2023)*.