Conditional Neural Heuristic for Multi-objective Vehicle Routing Problems

Mingfeng Fan, Yaoxin Wu, Zhiguang Cao, Wen Song, Guillaume Sartoretti, Huan Liu, and Guohua Wu*

***Abstract*—Existing neural heuristics for multi-objective vehicle routing problems (MOVRPs) are primarily conditioned on *instance context*, which failed to appropriately exploit *preference* and *problem size*, thus holding back the performance. To thoroughly unleash the potential, we propose a novel conditional neural heuristic (CNH) that fully leverages the instance context, preference, and size with an encoder-decoder structured policy network. Particularly, in our CNH, we design a dual-attention-based encoder to relate preferences and instance contexts, so as to better capture their joint effect on approximating the exact Pareto front (PF). We also design a size-aware decoder based on the sinusoidal encoding to explicitly incorporate the problem size into the embedding, so that a single trained model could better solve instances of various scales. Besides, we customize the REINFORCE algorithm to train the neural heuristic by leveraging stochastic preferences, which further enhances the training performance. Extensive experimental results on random and benchmark instances reveal that our CNH could achieve favorable approximation to the whole PF with higher hypervolume (HV) and lower optimality gap (Gap) than those of existing neural and conventional heuristics. More importantly, a single trained model of our CNH can outperform other neural heuristics that are exclusively trained on each size. Additionally, the effectiveness of the key designs is also verified through ablation studies.**

***Index Terms*—Encoder-decoder, Neural heuristic, Multi-objective optimization, Vehicle routing problems.**

I. Introduction

EXTENDING single-objective vehicle routing problems (SOVRPs) to encompass multiple conflicting criteria, multi-objective vehicle routing problems (MOVRPs) have garnered significant research interest in operations research and computer science, owing to their role in optimizing logistical and operational decision-making. These problems encapsulate a range of practical requirements in route optimization [1], balancing multiple goals such as cost efficiency, time management, and customer satisfaction, while also integrating considerations of environmental sustainability. Typical SOVRPs, such as the traveling salesman problem (TSP) and capacitated vehicle routing problem (CVRP), are well-known to be NP-hard. In comparison, MOVRPs are much harder [2], [3], as they aim to search for all Pareto optimal solutions (i.e. Pareto set) under different preferences in multiple objectives [4]. Due to their high computational complexity, various heuristics [4], [5] have been proposed to efficiently solve MOVRPs and pursue approximate Pareto optimal solutions. However, conventional heuristics are often specialized for each problem, lacking the ability to quickly adapt to other problems [6], [7]. Additionally, these heuristics are mostly hand-crafted and require massive manual work, thus potentially undermining their performance.

Recent years have seen a surge of attempts to solve SOVRPs using deep reinforcement learning (DRL) in a data-driven manner [8]–[10], most of which leverage encoder-decoder neural networks to automatically learn heuristics for vehicle routing problems (VRPs), e.g., TSP and CVRP [11]–[14]. These approaches, termed *neural heuristics*, can be trained to extract patterns and efficiently infer solutions on a collection of VRP instances [15], potentially eschewing laborious hand-crafted designs for each problem. Although extensive neural heuristics for SOVRPs have been investigated [16], there are only a few for MOVRPs. Typically, they decompose an MOVRP into a series of SOVRPs with different preferences, and design learning-based schemes to train deep model(s) for solving each resulting SOVRP [17]–[19].

Despite the recent advancements and success, existing neural heuristics for MOVRPs are still less effective. On the one hand, they process the *preference* (i.e. the relative importance or weights assigned to different objectives) independently of the *instance context* (i.e. the instance information including node locations, customer demands, and the vehicle capability). However, for each instance, a desirable Pareto front (PF) should be the one that is derived by jointly leveraging both the preference and the instance context, especially given that the same set of preferences could lead to different PFs for different instances. Hence, the instance-preference interaction in a more informative manner is vital to capture the exact PFs. On the other hand, they often struggle with training efficiency across multiple problem sizes and underestimate the effect of problem size in aid of learning more accurate objective landscapes of SOVRPs under each preference, thereby delivering inferior PFs of the MOVRP. Therefore, it is considerably rewarding to further ameliorate the performance by explicitly incorporating the size, so that a single unified model could be within reach and also outperform the ones trained for each size.

To address the above issues, in this paper, we propose a novel Conditional Neural Heuristic (CNH) for MOVRPs.

Mingfeng Fan, Huan Liu and Guohua Wu are with School of Traffic and Transportation Engineering, Central South University, China (E-mails: mingfan@csu.edu.cn, liuhuan1095@csu.edu.cn, guohuawu@csu.edu.cn).

Yaoxin Wu is with the Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Netherlands. (E-mail: wyxacc@hotmail.com)

Zhiguang Cao is with the School of Computing and Information Systems, Singapore Management University, Singapore. (E-mail: zhiguangcao@outlook.com)

Wen Song is with the Institute of Marine Science and Technology, Shandong University, China. (E-mail: wensong@email.sdu.edu.cn)

Guillaume Sartoretti is with the Department of Mechanical Engineering, National University of Singapore, Singapore (E-mail: guillaume.sartoretti@nus.edu.sg).

Corresponding authors: Guohua Wu.

Our CNH decomposes an MOVRP into a series of SOVRPs using a set of preferences and solves them in parallel with an encoder-decoder structured neural network to approximate the PF. In particular, the proposed encoder-decoder architecture is conditioned on the *instance context*, *preference*, and *problem size*, which collaboratively affect the PF and thus facilitate learning more informative policies for solving MOVRPs. In the encoder, we introduce a dual-attention to effectively relate the instance context and preference, which exploits a self-attention to promote representations of VRP nodes and the preference node under a specific SOVRP, as well as a cross-attention to emphasize the difference between the two heterogeneous nodes. In the decoder, we introduce a sinusoidal encoding to explicitly yield the problem size embeddings so as to allow the neural heuristic to be more generalizable for the route construction of different scales. With the instance context and preference processed by the encoder for better representation, the decoder iteratively selects a VRP node at each time step until a complete solution for a specific SOVRP is constructed. Moreover, we train our CNH on MOVRP instances of varying sizes using reinforcement learning in conjunction with stochastic preferences to boost performance. We evaluate our CNH on randomly generated instances of TSP with two and three objectives and CVRP with two objectives, as well as benchmark instances, and the results show that it achieves better PFs than those of conventional and neural baselines for all problem sizes either seen or unseen during training.

The remainder of this paper is organized as follows: Section II reviews related works; Section III presents the formulation of MOVRPs and different decomposition strategies; Section IV details the proposed method and policy network; Section V exhibits and analyzes our experimental results; Section VI identifies limitations of our method; Section VII concludes this paper.

II. Related Works

A. Conventional Methods for MOVRPs

The classic methods for MOVRPs are mainly divided into two branches, i.e., exact algorithms and heuristics. The former ones aim to generate all exact Pareto optimal solutions, which inevitably suffer the exponentially growing complexity and thus the unbearable computational cost, especially for large-sized problems [20]. Consequently, the latter ones are often used in practice [21]–[23], since the heuristics are able to attain approximate PFs in a reasonable time, with the compromise on exactness. During the last two decades, multi-objective evolutionary algorithms (MOEAs), including the dominance-based and decomposition-based ones, have been recognized as the mainstream conventional methods [24]–[26] and proved superior in tackling MOVRPs [27]. In a typical MOEA, a working population of a predetermined size is evolved using reproduction, mutation, and selection operators. The selection operator is usually designed based on Pareto dominance, aggregation (i.e. combining the multiple objectives into a single scalar objective function based on a decomposition strategy), or some other performance indicators. Then, MOEAs may use another population, referred to as the archive, to store nondominated solutions generated during the evolutionary process [28]. Although many MOEA variants have been proposed [29]–[31], they are still lacking in that they always need numerous iterations for solving large-scale problems with low computational efficiency. Moreover, most of them are hand-crafted for specific problems with much manual effort, which may hold back the performance.

B. Neural Heuristics for MOVRPs

Motivated by the recent success in learning data-driven heuristics for SOVRPs (e.g. the attention model (AM) [11] and policy optimization with multiple optima (POMO) [32]), growing attention has been paid to explore neural heuristics for solving MOVRPs. Li et al. [18] and Wu et al. [19] decompose an MOVRP into multiple scalarized subproblems (i.e. SOVRPs), and train separate models for each subproblem with deep reinforcement learning (DRL) and parameter transfer between models. Instead of DRL, Shao et al. [33] and Zhang et al. [34] employ evolutionary algorithms to update or fine-tune parameters in neural networks for solving each SOVRP and pursuing non-dominant solutions. Zhang et al. [2] propose a meta-learning-based DRL method, which evolves the meta-model to a number of sub-models via the fine-tuning process. All the above neural heuristics rely on multiple deep models to solve MOVRPs, which are likely to incur prohibitive training overhead. In contrast, Lin et al. [17] propose a single decomposition-based model with DRL to solve MOVRPs, where they use hypernetwork to process the preferences in a decoder so as to intervene the parameter training and route construction for each SOVRP. Termed PMOCO, this method is recognized as the state-of-the-art neural heuristic for MOVRPs. Although the performance is favorable against conventional heuristics, PMOCO is still suffering from two drawbacks: 1) PMOCO mainly processes the preferences in the decoder, which is independent of the instance context, thus less effective in approximating the exact PF; 2) PMOCO trains a deep model for each problem size, which is less practical while ignoring the effect of size in yielding high-quality solutions. Different from PMOCO, in this paper, we propose a novel neural heuristic conditioned on the instance context, preference and problem size, which exceeds other heuristics with only a single trained model.

III. Preliminaries

A. Problem Statement

A VRP (or SOVRP) instance can be defined on a graph with nodes $V=\{0,1,2,\cdots,n\}$, where a node $i\in V$ 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 $\pi_j\in V$. A solution π is feasible only if it meets the constraints for the VRP. Accordingly, a VRP with m objectives (i.e. MOVRP) is formally defined as

$$\min_{\pi\in\mathscr{X}} F(\pi)=(f_1(\pi),f_2(\pi),\cdots,f_m(\pi)), \tag{1}$$

where $\mathscr{X}$ comprises all feasible solutions to the MOVRP, and $F(\pi)$ is a m-dimensional vector containing m objective values

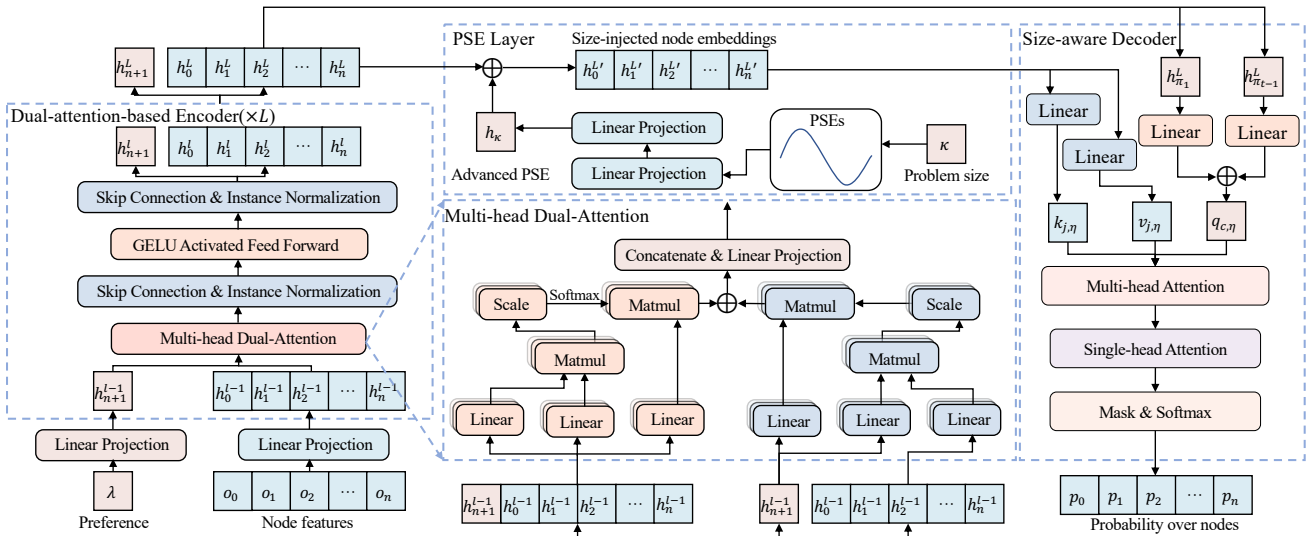


Fig. 1. An illustration of our policy network for a specific SOVRP instance with $n+1$ nodes and a preference vector λ. For the given SOVRP instance, the features of nodes along with the preference vector are first processed through a dual-attention-based encoder, in which the node embeddings $\{h_i^{l-1}\}_{i=0}^n$ and the preference embedding h_{n+1}^{l-1} are interacted by the multi-head dual-attention module. After the node embeddings $\{h_i^L\}_{i=0}^n$ are derived from the encoder, the problem size κ ($\kappa = n+1$) is fed into the problem size embedding (PSE) layer to produce an advanced problem size embedding (PSE), which is then combined with $\{h_i^L\}_{i=0}^n$ to create the size-injected node embeddings $\{h_i^{L'}\}_{i=0}^n$. The size-aware decoder, starting with an empty sequence, iteratively selects a node at each step to form a complete solution. In each time step, it utilizes both the node embeddings and the size-injected node embeddings to generate a probability vector over the nodes with masked softmax, from which it selects the next node to visit. This process continues until a full solution sequence is constructed.

of the solution π. In the cases of conflicting objectives, there is no single solution achieving optimality for every objective. Instead, Pareto optimal solutions are often pursued to capture or reflect various trade-offs under different preferences over the objectives.

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

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

Since SOVRPs (e.g. TSP and CVRP) are already NP-hard, MOVRPs are more intractable to attain Pareto optimal solutions, the quantity of which often exponentially expands along with the problem size (i.e. the number of nodes). Here we elaborate the problem definitions for two typical MOVRP variants studied in this paper, i.e., multi-objective TSP (MOTSP) and multi-objective CVRP (MOCVRP).

MOTSP. An MOTSP instance is gauged by multiple cost matrices (objectives), with the aim of finding a group of tours, i.e., node sequences, which are Pareto optimal. For example, if an m-objective TSP instance s with $n+1$ nodes is measured by cost matrices $C^i = (c^i_{jk})$, with $i \in \{1,\cdots,m\}$ and $j,k \in \{0,\cdots,n\}$, then the ith objective is defined as

$$f_i(\pi) = c^i_{\pi_{n+1}\pi_1} + \sum_{j=1}^{n} c^i_{\pi_j \pi_{j+1}}, \tag{2}$$

where $\pi = (\pi_1, \pi_2, \cdots, \pi_{n+1})$ with $\pi_j \in \{0,1,\cdots,n\}$ and each node is accessed exactly once. We consider the Euclidean MOTSP following [17], [18], [20]. Each node j has a $2m$-dimensional feature vector $[o_j^1, o_j^2, \cdots, o_j^m]$, where $o_j^i \in \mathbb{R}^2$ is the coordinate under the i-th objective. Thus, the objective $f_i(\pi)$ can also be expressed as $f_i(\pi) = \|o^i_{\pi_{n+1}} - o^i_{\pi_1}\|_2 + \sum_{j=1}^{n} \|o^i_{\pi_j} - o^i_{\pi_{j+1}}\|_2$. In this paper, we consider MOTSP with both two and three objectives.

MOCVRP. An MOCVRP instance consists of n customer nodes with coordinates loc_i ($i \in \{1,\cdots,n\}$) and demands δ_i ($\delta_0 > 0$), and a depot node with loc_0 and δ_0 (δ_0=0). A vehicle with the capacity $\mathscr{Q}$ ($\mathscr{Q} > \delta_i$) serves all customers in multiple routes (starting and ending at the depot), which must ensure that 1) each customer is visited exactly once; 2) the total demand of customers in each route should not exceed the vehicle capacity. Following existing literature on neural heuristics [17], [35], we consider bi-objective CVRP in this paper. Particularly, the objectives are to minimize the total length of all routes and the length of the longest route.

B. Decomposition Strategy

Decomposition is a simple yet efficient strategy for MOVRPs and has been widely applied in MOEAs [25]. It scalarizes an MOVRP into subproblems (SOVRPs) under different preferences, which are then solved to pursue the Pareto optimal solutions. Given a preference denoted as $\lambda \in \mathbb{R}^m$ with $\lambda_i \geq 0$ and $\sum_{i=1}^m \lambda_i = 1$, two typical decomposition strategies including Weighted-sum and Tchebycheff [36] are usually exploited.

Weighted-sum Decomposition. It minimizes the convex combination of m objectives under each preference as

$$\min_{\pi \in \mathscr{X}} g_w(\pi|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(\pi). \tag{3}$$

Tchebycheff Decomposition. It minimizes the maximal distance between objectives and the ideal point as

$$\min_{\pi\in\mathscr{X}} g_t(\pi|\lambda) = \max_{1\le i\le m} \{\lambda_i |f_i(\pi) - (z_i^* - \varepsilon)|\}, \tag{4}$$

where λ is the preference, z_i^* is the ideal value for objective $f_i(\pi)$ and ε is a small positive component. While Weighted-sum cannot be used for nonconvex PFs, Tchebycheff has no such limitation. In fact, the optimal solution in Eq. (4) under a specific (but unknown) preference λ could be a Pareto optimal solution [37]. Hence, in this paper, we use Tchebycheff decomposition to scalarize MOVRPs, same as [17].

Typically, after decomposition, conventional heuristics will solve each SOVRP with hand-crafted local search or evolutionary algorithms to approximate the PF. However, they often deliver a limited number of or inferior approximate solutions in practice. Different from them, we propose a data-driven conditional neural heuristic (CNH) to solve the resulting SOVRPs in parallel for achieving better approximate performance with respect to the Pareto front.

IV. Methodology

A. Overview

The decomposition strategy provides a paradigm to pursue the Pareto set for an MOVRP, by solving SOVRPs derived from different preferences. To achieve high-quality approximate Pareto optimal solutions, we propose a neural heuristic conditioned on the instance context, preference, and problem size. Given an MOVRP instance s (or its instance context) with the problem size κ and a preference λ, our CNH is designed to learn a stochastic policy p_θ for obtaining the approximate Pareto solution $\pi = (\pi_1, \pi_2, \cdots, \pi_T)$, where θ represents the set of learnable parameters within our neural architecture, including weights and bias in both encoder and decoder, expressed as,

$$p_\theta(\pi|s,\lambda,\kappa) = \prod_{t=1}^{T} p_\theta(\pi_t|s,\lambda,\kappa,\pi_{1:t-1}), \tag{5}$$

where π_t and $\pi_{1:t-1}$ represent the selected node and partial solution at time step t, and T denotes the number of steps to construct the solution π. To this end, we design an encoder-decoder neural architecture, where besides the instance context fed to the encoder as default, we also elegantly incorporate the preference and problem size to the encoder and decoder, respectively. Accordingly, on the one hand, we propose a dual attention in the encoder to embed both the instance context and preference, and also relate them to identify their joint effect on the solutions to SOVRPs, thus the eventual PFs. On the other hand, we design a size-aware decoder that yields problem size embeddings with the sinusoidal encoding to intervene the policy learning over a wide range of scales. The neural architecture of our CNH is illustrated in Fig. 1, and we customize the REINFORCE algorithm to train it with different-sized MOVRP instances and stochastic preferences. The key designs are elaborated as follows.

B. Dual-attention-based Encoder

Given an instance s (or its context) with nodes $V = \{0,1,2,\cdots,n\}$ and their features $\{o_i\}_{i=0}^n$ (e.g. coordinates in MOTSP), the encoder takes the instance s and a preference λ as inputs to yield the embeddings which capture their relationship. In specific, we first transform $\{o_i\}_{i=0}^n$ and λ into initial node (instance context) embeddings $\{h_i^0\}_{i=0}^n$ and preference embedding h_λ^0 using separate linear projections with the same output dimension d_h=128. The vanilla self-attention layer treats all inputs in a homogeneous fashion [38]. However, the preference and instance nodes (or context) are obviously of different types. To discriminate their embeddings, we design a dual-attention layer that comprises a multi-head dual-attention (MDA) sublayer and a feed-forward (FF) sublayer. We evolve the initial embeddings $\{h_i^0\}_{i=0}^n$ and h_λ^0 through this layer L times, deriving more informative embeddings $\{h_i^L\}_{i=0}^n$ and h_λ^L.

MDA Sublayer. Given the node embeddings and preference embedding $\{h_i^{l-1}\}_{i=0}^n$ and h_λ^{l-1} as inputs to l-th dual-attention layer, we first define h_{n+1}^{l-1}=h_λ^{l-1} (for expression simplicity) and compute the multi-head self-attention scores between the nodes and preference as follows,

$$a_{i,j,\eta}^s = \frac{1}{\sqrt{d_k}}(h_i^{l-1}W_\eta^{Q_s})(h_j^{l-1}W_\eta^{K_s})^\top, \quad \forall i,j \in \{0,\cdots,n+1\} \tag{6}$$

where $W_\eta^{Q_s} \in \mathbb{R}^{d_h\times d_q}$ and $W_\eta^{K_s} \in \mathbb{R}^{d_h\times d_k}$ are trainable *query* and *key* matrices for the head η ($\eta \in \{1,\cdots,H\}$), with $H=8$ and d_q=d_k=d_h/H. The attention scores $\{a_{i,j,\eta}^s\}_{j=0}^{n+1}$ are normalized to $\{\tilde{a}_{i,j,\eta}^s\}_{j=0}^{n+1}$ over the index j using the Softmax function, which are then used to calculate self-attention values $\{h_{i,\eta}^s\}_{i=0}^{n+1}$ in every head as follows,

$$h_{i,\eta}^s = \sum_{j=0}^{n+1} \tilde{a}_{i,j,\eta}^s (h_j^{l-1}W_\eta^{V_s}), \forall i \in \{0,\cdots,n+1\} \tag{7}$$

where $W_\eta^{V_s} \in \mathbb{R}^{d_h\times d_k}$ is the trainable *value* matrix. Then, we concatenate self-attention values in different heads to obtain the aggregated node and preference embeddings $\{h_i^s\}_{i=0}^{n+1}$ with d_h dimension, i.e., $h_i^s = \left[h_{i,1}^s;\cdots;h_{i,H}^s\right]$.

The multi-head self-attention views the preference as an extra (dummy) node and processes it with those genuine nodes homogeneously. To better discriminate them and capture their joint effect on the solutions (thus the eventual PF), we further exploit a multi-head cross-attention to yield another group of embeddings $\{h_i^c\}_{i=0}^n$. The corresponding scores are computed as follows,

$$a_{i,\eta}^c = \frac{1}{\sqrt{d_k}}(h_i^{l-1}W_\eta^{Q_c})(h_{n+1}^{l-1}W_\eta^{K_c})^\top, \ \forall i \in \{0,\cdots,n\} \tag{8}$$

where $W_\eta^{Q_c} \in \mathbb{R}^{d_h\times d_q}$ and $W_\eta^{K_c} \in \mathbb{R}^{d_h\times d_k}$ are trainable query and key matrices for the head η. Then we compute the cross-attention values $\{h_{i,\eta}^c\}_{i=0}^n$ in each head as follows,

$$h_{i,\eta}^c = a_{i,\eta}^c (h_{n+1}^{l-1}W_\eta^{V_c}), \forall i \in \{0,\cdots,n\} \tag{9}$$

where $W_\eta^{V_c} \in \mathbb{R}^{d_h\times d_k}$ is the trainable value matrix for the head η. Accordingly, we concatenate the values in different heads

to attain the aggregated node embeddings $\{h_i^c\}_{i=0}^n$ out of the multi-head cross-attention, i.e., $h_i^c = \left[h_{i,1}^c; \cdots; h_{i,H}^c\right]$.

To fuse the node (instance context) embeddings and preference embeddings derived from the dual mechanisms, i.e., the multi-head self-attention and the multi-head cross-attention, we directly add them as follows,

$$h_i^{l'} = \begin{cases} h_i^s + h_i^c, & \forall i \in \{0, \cdots, n\} \\ h_i^s, & i = n+1 \end{cases} \tag{10}$$

which are then successively processed using linear projection, skip-connection, and instance normalization (IN) [39], [40], i.e., $h_i^{l'} = \mathrm{IN}(h_i^{l-1} + h_i^{l'} W_o)$, with a trainable matrix W_o. Note that in the multi-head cross-attention mechanism, we do not utilize preference embedding to attend to node embeddings, as our experiments (see Section V-F) have shown that this does not contribute to the improved performance. Instead, we set the preference embedding to zeros.

FF Sublayer. The embeddings $\{h_i^{l'}\}_{i=0}^{n+1}$ from the MDA sublayer are taken as inputs to a feed-forward sublayer, which consists of two linear layers with the GELU activation function in between [41], such that,

$$h_i^f = \mathrm{GELU}(h_i^{l'} W_1) W_2, \ \forall i \in \{0, \cdots, n+1\} \tag{11}$$

where $W_1 \in \mathbb{R}^{d_h \times d_f}$ and $W_2 \in \mathbb{R}^{d_f \times d_h}$ are trainable matrices with $d_f = 512$. We process $\{h_i^f\}_{i=0}^{n+1}$ using skip-connection and IN to obtain the output embeddings $\{h_i^l\}_{i=0}^{n+1}$ from the l-th dual-attention layer as follows,

$$h_i^l = \mathrm{IN}(h_i^{l'} + h_i^f), \ \forall i \in \{0, \cdots, n+1\} \tag{12}$$

We stack L ($L = 6$) dual-attention layers in the encoder, each of which possesses the same structure but with respective parameters. The embeddings of the (genuine) nodes from the encoder, i.e., $\{h_i^L\}_{i=0}^n$, are adopted as inputs to the decoder.

C. Size-aware Decoder

In the decoder, we first feed the number of nodes to the problem size embeddings (PSE) layer, and evolve the embeddings through a multi-head attention (MHA) layer, followed by a single-head attention (SHA) layer to select nodes for route construction (as expressed in Eq. (5)).

PSE Layer. We employ the sinusoidal encoding based on sine and cosine functions to yield the problem size embeddings (PSEs), which have been proven effective for positional encoding of tokens in NLP tasks [38]. In specific, we define the initial PSEs as follows,

$$\begin{aligned} \mathrm{PSE}(\kappa, 2i) &= sin(\kappa / 10000^{2i/d_h}), \\ \mathrm{PSE}(\kappa, 2i+1) &= cos(\kappa / 10000^{2i/d_h}), \end{aligned} \tag{13}$$

where κ and i ($i \in \{0, \cdots, 63\}$) mean the problem size and dimension, respectively. The resulting 128-dimensional PSEs are then processed using two linear layers with trainable matrices $W_{\kappa 1} \in \mathbb{R}^{d_h \times 2d_h}$ and $W_{\kappa 2} \in \mathbb{R}^{2d_h \times d_h}$. We inject the size information by adding the results from PSE to $\{h_i^L\}_{i=0}^n$ from the encoder such that,

$$h_i^{L'} = h_i^L + h_\kappa, \text{ with } h_\kappa = (\mathrm{PSE}(\kappa, \cdot) W_{\kappa 1}) W_{\kappa 2}. \tag{14}$$

The increment of the size-injected node embeddings $\{h_i^{L'}\}_{i=0}^n$ captures the context of problem size information and assists our CNH model in pursuing the best-performing strategy for a size-specific instance. In other words, the injection of problem size in CNH enables the extraction of more informative node embeddings, which facilitate the neural network to learn more high-quality PFs.

MHA Layer. At time step t in the solution construction, we calculate the context embedding based on the first and last visited node in the current partial solution $\pi_{1:t-1}$, as follows,

$$q_{c,\eta} = h_{\pi_1}^L W_\eta^{Q_1} + h_{\pi_{t-1}}^L W_\eta^{Q_t}, \tag{15}$$

where $W_\eta^{Q_1}, W_\eta^{Q_t} \in \mathbb{R}^{d_h \times d_q}$ are trainable matrices. We regard the context embedding $q_{c,\eta}$ as a query to attend to all nodes. To this end, we derive the key and value vectors from size-injected node embeddings $h_j^{L'}$ (rather than h_j^L) such that,

$$k_{j,\eta} = h_j^{L'} W_\eta^K; \ v_{j,\eta} = h_j^{L'} W_\eta^V, \ \forall j \in \{0, \cdots, n\} \tag{16}$$

where $W_\eta^K, W_\eta^V \in \mathbb{R}^{d_h \times d_k}$ are trainable parameters. Then, we compute attention scores as $a_{j,\eta} = q_{c,\eta} k_{j,\eta}^\top / \sqrt{d_k}$, and mask infeasible nodes (by setting $a_{j,\eta} = -\infty$) which cannot be visited next (e.g. the already visited nodes for MOTSP). Afterwards, we normalize $\{a_{j,\eta}\}_{j=0}^n$ to $\{\tilde{a}_{j,\eta}\}_{j=0}^n$ using Softmax function and obtain the attention value as $u_{c,\eta}$ in the head η. We aggregate these values through a linear layer (with trainable $W_3 \in \mathbb{R}^{d_k \times d_h}$) to update the context embedding, such that,

$$\tilde{q}_c = \sum_{\eta=1}^{H} u_{c,\eta} W_3, \text{ with } u_{c,\eta} = \sum_{j=0}^{n} a_{j,\eta} v_j. \tag{17}$$

SHA Layer. In the final stage, an SHA layer is employed to compute the probabilities of selecting each feasible node. In this layer, we compute the attention scores as below,

$$a_j = C \cdot \mathrm{Tanh}(\tilde{q}_c (h_j^{L'} W_4)^\top / \sqrt{d_k}), \forall j \in \{0, \cdots, n\} \tag{18}$$

where $W_4 \in \mathbb{R}^{d_h \times d_h}$ and Tanh function are used for linear projection and value clipping with C=10. Again, we mask infeasible nodes and normalize $\{a_j\}_{j=0}^n$ using the Softmax function to obtain the probabilities, based on which the next node to visit is selected.

During training, the size-aware decoder chooses actions (i.e. nodes) using a sampling strategy (i.e. sampling actions according to the probabilities derived from the decoder). During testing, the greedy strategy (i.e. choosing the action with the maximum probability) is used to construct the solution by the decoder. While the sampling is used during training for enhancing the RL exploration in solution space, the greedy decoding is used during testing to economize the time to reason out the solution.

D. Objective and Optimization

We expect to train the policy p_θ of our CNH that solves the respective SOVRPs to obtain approximate Pareto optimal solutions, which are conditioned on the instance context, preference, and problem size. Given an MOVRP instance as

Algorithm 1 Customized REINFORCE for CNH

Input: Instance distribution $\mathscr{S}_I$, preference distribution $\mathscr{S}_\lambda$, problem size distribution Λ, number of training epochs E, number of training instances per epoch $\mathbb{D}$, batch size B, number of tours N.

Output: The trained CNH (i.e. p_θ).

Initialize policy network parameters θ.
for $e = 1$ to E **do**
  $D = 0$,
  while $D < \mathbb{D}$ **do**
    $\tilde{B} = min(\mathbb{D} - D, B)$,
    $\kappa \sim$ SAMPLEPROBLEMSIZE(Λ),
    $\lambda_b \sim$ SAMPLEPREFERENCE ($\mathscr{S}_\lambda$), $\forall b \in \{1, \cdots, \tilde{B}\}$,
    $s_b \sim$ SAMPLEINSTANCE ($\mathscr{S}_I$), $\forall b \in \{1, \cdots, \tilde{B}\}$,
    $\pi_b^j \sim$ SAMPLETOUR($p_\theta(\cdot|s_b, \lambda_b, \kappa)$),
      $\forall j \in \{1, \cdots, N\}$, $\forall b \in \{1, \cdots, \tilde{B}\}$,
    Calculate gradient $\nabla J(\theta)$ according to Eq. (21),
    $\theta \leftarrow$ **ADAM**($\theta, \nabla J(\theta)$),
    $D = D + \tilde{B}$,
  end while
end for

expressed in Eq. (1), we define the reward for each subproblem using Tchebycheff decomposition as

$$R(\pi|s, \lambda, \kappa) = -\max_{1 \le i \le m} \{\lambda_i | f_i(\pi) - (z_i^* - \varepsilon)|\}. \quad (19)$$

With the rewards, we customize REINFORCE [42] to compute the gradient for maximizing the expected return $J(\theta) = E_{\pi \sim p_\theta, s \sim \mathscr{S}_I, \lambda \sim \mathscr{S}_\lambda, \kappa \sim \Lambda} R(\pi|s, \lambda, \kappa)$ such that,

$$\begin{aligned} \nabla J(\theta) = & E_{\pi \sim p_\theta, s \sim \mathscr{S}_I, \lambda \sim \mathscr{S}_\lambda, \kappa \sim \Lambda} [(R(\pi|s, \lambda, \kappa) \\ & - \tilde{R}(s, \lambda, \kappa)) \nabla_\theta \log p_\theta(\pi|s, \lambda, \kappa)], \end{aligned} \quad (20)$$

where $\tilde{R}(s, \lambda, \kappa)$ is the baseline of expected reward to reduce the variance of the sampled gradients. We use Monte Carlo sampling to approximate J by training the policy with minibatches of instances and sampling multiple tours for each SOVRP. In specific, given B instance-preference pairs $\{(s_b, \lambda_b)\}_{b=1}^B$ with problem size κ and N sampled tours $\{\pi_b^j\}_{j=1}^N$ for each SOVRP under (s_b, λ_b), we calculate the approximate gradient as below,

$$\begin{aligned} \nabla J(\theta) \simeq & \frac{1}{BN} \sum_{b=1}^{B} \sum_{j=1}^{N} (R(\pi_b^j|s_b, \lambda_b, \kappa) \\ & - \tilde{R}(s_b, \lambda_b, \kappa)) \nabla_\theta \log p_\theta(\pi_b^j|s_b, \lambda_b, \kappa), \end{aligned} \quad (21)$$

where we set $\tilde{R}(s_b, \lambda_b, \kappa) = \frac{1}{N} \sum_{j=1}^N R(\pi_b^j|s_b, \lambda_b, \kappa)$ and force different starting nodes to sample the N tours for each SOVRP following [32]. The pseudocode of the optimization procedure is summarized in Algorithm 1.

Size Sampling. To strengthen the capability of our CNH in solving different-sized MOVRPs, we train it with instances of multiple sizes at the same time. In specific, we sample a size from a predefined set Λ for each minibatch and then generate training instances of the corresponding size for optimization as in Eq. (21).

TABLE I
TRAINING TIME ON BI-CVRP.

Neural heuristics	Bi-CVRP20	Bi-CVRP50	Bi-CVRP100
DRL-MOA	18.50h	46.12h	65.93h
ML-DAM	428.32h	420.51h	124.18h
PMOCO	7.78h	13.08h	42.38h
CNH		20.31h	

Stochastic Preferences. Existing neural heuristics restrict the sum of values in preferences to 1, i.e., $\sum_{i=1}^m \lambda_i = 1$ [17], [18]. However, it is clear that preferences like $[0.1, 0.4]$ and $[0.2, 0.8]$ share the same meaning since these two vectors would represent the same directions if they are normalized (so that the sum of the values in each vector should be equal to 1). Moreover, preference vectors that point in the same direction will lead the optimization toward the same region of the Pareto front. Hence we generate more stochastic preferences during training by randomly sampling each value in preferences within [0, 1], which are taken as input to the neural network. When computing the rewards, we still scale preferences to ensure normalization, i.e., the sum of all preference values is equal to 1. We may view the stochastic preferences as a way of enhancing the training data diversity, since preferences with the same/similar orientations may share the same/similar solutions and rewards.

V. EXPERIMENTS

A. Setup

Problems&Training. We conduct extensive experiments to verify the effectiveness of our CNH on bi-objective TSP (Bi-TSP), tri-objective TSP (Tri-TSP), and bi-objective CVRP (Bi-CVRP), which are used in most literature of neural heuristics [2], [17], [43]. For the m-objective TSP, each node i is featured by m 2D-coordinates and the m-th cost from node i to j is the Euclidean distance between their m-th 2D-coordinates. For the Bi-CVRP, the conflicting objectives are to minimize the total tour length and the length of the longest route, following [5], [35]. We use the same hyperparameters for all problems and train a single CNH for each problem (rather than training models for each problem size in existing neural heuristics). We generate 100,000 instances on the fly in each epoch, with 2D-coordinates and demands uniformly sampled from $[0, 1]^2$ and the set $\{1, \cdots, 9\}$, respectively. We set the batch size to 64, and randomly sample the problem size from $\Lambda = [20, 40, 60, 80, 100]$, which means the number of graph nodes and customer nodes in MOTSP and MOCVRP, respectively. We train CNH for 200 epochs with Adam optimizer and set the learning rate and weight decay to 10^{-4} and 10^{-6}, respectively.

Baselines. We compare our CNH with four widely-used MOEAs and three neural heuristics for MOVRPs, including 1) **MOEA/D** [25], a classic decomposition-based MOEA implemented with 4,000 iterations on Pymoo [1]; 2) **NSGA-II** [24], a typical Pareto dominance-based multi-objective

[1] https://pymoo.org/

TABLE II
RESULTS ON MOVRPS WITH DIFFERENT PROBLEM SIZES.

Method	Bi-TSP20 HV	Gap	Time	Bi-TSP50 HV	Gap	Time	Bi-TSP100 HV	Gap	Time
MOEA/D	0.508	0.45%	8.41h	0.552	3.91%	12.81h	0.587	12.80%	19.69h
NSGA-II	0.509	0.19%	3.99h	0.561	2.45%	8.71h	0.579	14.04%	16.36h
NSGA-III	0.508	0.34%	4.25h	0.546	5.05%	9.02	0.563	16.38%	16.91
MOGLS	0.509	0.20%	2.11h	0.522	9.17%	6.60h	0.556	17.34%	17.16h
DRL-MOA	0.434	14.94%	1.75s	0.491	14.53%	3.60s	0.608	9.76%	7.42s
ML-DAM	0.505	1.06%	2.46s	0.555	3.51%	4.57s	0.645	4.17%	8.69s
PMOCO	0.508	0.49%	4.16s	0.567	1.27%	7.46s	0.666	1.01%	22.12s
PMOCO-Aug	**0.510**	**0.00%**	1.29m	0.575	0.03%	4.29m	0.673	0.10%	19.06m
CNH	0.508	0.37%	5.13s	0.569	0.97%	8.34s	0.668	0.74%	24.71s
CNH-Aug	0.510	0.02%	1.34m	**0.575**	**0.00%**	4.54m	**0.673**	**0.00%**	19.73m

Method	Bi-CVRP20 HV	Gap	Time	Bi-CVRP50 HV	Gap	Time	Bi-CVRP100 HV	Gap	Time
MOEA/D	0.226	0.81%	12.32h	0.259	1.99%	21.75h	0.232	7.49%	36.62h
NSGA-II	**0.227**	0.30%	7.65h	0.260	1.44%	17.34h	0.208	16.93%	34.34h
NSGA-III	0.228	0.00%	8.25	0.260	1.47%	18.36h	0.209	16.52%	35.80h
MOGLS	0.226	0.60%	7.09h	0.254	4.14%	14.52h	0.213	14.78%	45.47h
DRL-MOA	0.180	20.88%	6.42s	0.226	16.27%	10.36s	0.213	14.82%	19.87s
ML-DAM	0.211	7.21%	3.06s	0.223	17.63%	5.76s	0.191	23.64%	10.93s
PMOCO	0.220	3.39%	5.40s	0.259	1.81%	11.08s	0.207	17.17%	27.10s
PMOCO-Aug	0.223	2.11%	17.27s	0.262	0.66%	43.35s	0.218	12.82%	3.03m
CNH	0.223	2.07%	6.87s	0.262	0.57%	10.02s	0.249	0.44%	30.84s
CNH-Aug	0.225	1.28%	17.38s	**0.263**	**0.00%**	44.32s	**0.250**	**0.00%**	3.25m

Method	Tri-TSP20 HV	Gap	Time	Tri-TSP50 HV	Gap	Time	Tri-TSP100 HV	Gap	Time
MOEA/D	0.203	37.83%	9.98h	0.184	47.44%	14.13h	0.230	48.95%	21.61h
NSGA-II	0.285	13.04%	4.04h	0.169	51.48%	9.04h	0.156	65.41%	16.87h
NSGA-III	0.308	5.95%	4.78h	0.253	27.60%	9.73h	0.258	42.72%	18.18h
MOGLS	0.323	1.44%	4.58h	0.291	16.70%	14.56h	0.323	28.38%	37.96h
DRL-MOA	0.250	23.77%	3.51s	0.235	32.67%	5.51s	0.353	21.76%	9.50s
ML-DAM	0.294	10.24%	3.24s	0.282	19.16%	5.74s	0.337	25.33%	9.45s
PMOCO	0.324	1.13%	4.40s	0.339	2.81%	7.67s	0.441	2.39%	23.03s
PMOCO-Aug	**0.327**	**0.00%**	1.28m	0.348	0.37%	4.50m	0.447	0.86%	19.90m
CNH	0.326	0.49%	5.71s	0.345	1.35%	8.91s	0.446	1.11%	23.84s
CNH-Aug	0.327	0.00%	1.38m	**0.349**	**0.00%**	4.82m	**0.451**	**0.00%**	20.41m

TABLE III
RESULTS ON LARGER-SCALE MOVRPS.

Method	Bi-TSP150 HV	Gap	Time	Bi-TSP200 HV	Gap	Time
MOEA/D	0.584	18.93%	26.43h	0.575	23.12%	34.23h
NSGA-II	0.547	24.08%	22.74h	0.510	31.78%	31.52h
NSGA-III	0.535	25.70%	23.72h	0.502	32.93%	32.08h
MOGLS	0.564	21.75%	48.96h	0.563	24.66%	74.56h
DRL-MOA	0.658	8.68%	17.63s	0.688	8.01%	22.74s
ML-DAM	0.694	3.71%	12.47s	0.722	3.42%	17.98s
PMOCO	0.714	0.93%	1.01m	0.741	0.98%	2.02m
PMOCO-Aug	0.719	0.21%	59.18m	0.745	0.33%	2.05h
CNH	0.716	0.62%	1.02m	0.744	0.55%	2.02m
CNH-Aug	**0.720**	**0.00%**	59.44m	**0.748**	**0.00%**	2.08h

Method	Bi-CVRP150 HV	Gap	Time	Bi-CVRP200 HV	Gap	Time
MOEA/D	0.229	16.16%	54.97h	0.192	32.86%	68.38h
NSGA-II	0.206	24.44%	51.49h	0.189	33.86%	69.56h
NSGA-III	0.207	24.18%	50.17h	0.190	33.48%	68.57h
MOGLS	0.233	14.71%	74.75h	0.238	16.72%	97.42h
DRL-MOA	0.239	12.60%	33.82s	0.245	14.02%	47.11s
ML-DAM	0.166	39.23%	15.13s	0.209	26.73%	21.54s
PMOCO	0.215	21.17%	1.18m	0.219	23.27%	2.42m
PMOCO-Aug	0.230	15.82%	8.72m	0.236	17.17%	18.20m
CNH	0.272	0.26%	1.23m	0.285	0.18%	2.44m
CNH-Aug	**0.273**	**0.00%**	9.01m	**0.285**	**0.00%**	18.86m

Method	Tri-TSP150 HV	Gap	Time	Tri-TSP200 HV	Gap	Time
MOEA/D	0.253	49.58%	28.23h	0.266	50.18%	35.44h
NSGA-II	0.136	72.96%	25.44h	0.119	77.68%	33.91h
NSGA-III	0.235	53.13%	25.63h	0.208	61.14%	34.06h
MOGLS	0.329	34.57%	70.27h	0.328	38.68%	109.52h
DRL-MOA	0.410	18.47%	20.83s	0.447	16.38%	25.51s
ML-DAM	0.384	23.51%	13.95s	0.416	22.16%	19.93s
PMOCO	0.495	1.57%	1.05m	0.529	1.10%	2.10m
PMOCO-Aug	0.501	0.36%	1.04h	0.534	0.07%	2.13h
CNH	0.499	0.66%	1.08m	0.532	0.52%	2.11m
CNH-Aug	**0.502**	**0.00%**	1.05h	**0.535**	**0.00%**	2.15h

genetic algorithm, also implemented with 4,000 iterations on Pymoo; 3) **NSGA-III** [44], an extension of NSGA-II by introducing reference direction, which is implemented with 4,000 iterations on Pymoo; 4) **MOGLS** [45], a multi-objective genetic local search algorithm implemented with 10,000 iterations and 100 local search steps in each iteration [2]; 5) **DRL-MOA** [18], which decomposes an MOVRP into SOVRPs under different preferences and solves them with a parameter transfer scheme[3]; 6) **ML-DAM** [2], which fine-tunes a learned meta-model to deliver multiple models for respective SOVRPs[4]; 7) **PMOCO** [17], a preference-based hypernetwork which achieves state-of-the-art performance in MOTSP and MOCVRP among the present neural heuristics[5]. Regarding MOEAs (i.e. MOEA/D, NSGA-II, NSGA-III, and MOGLS), we use 2-opt and problem-specific local search for solving MOTSP and Bi-CVRP, respectively.

Training Efficiency. We train DRL-MOA, ML-DAM, and PMOCO on problem sizes 20, 50, and 100 for all problem types. Among them, DRL-MOA and PMOCO are trained using their original settings. Pertaining to ML-DAM, we set the number of iterations to 10,000, 5,000, and 1,000 for Bi-CVRP20, Bi-CVRP50, and Bi-CVRP100, respectively, while keeping the other training settings the same as in the original paper. This small change promotes a fair comparison by ensuring that ML-DAM costs similar training overhead to other baselines, otherwise its training time could be prohibitively long due to the meta-learning scheme. We record the training time of the four neural heuristics (i.e. DRL-MOA, ML-DAM, PMOCO, and CNH) on Bi-CVRP in TABLE I. Our results show that DRL-MOA and ML-DAM require significantly longer training time than CNH for most problem sizes, as they need to train separate models for each SOVRP. On the other hand, while PMOCO takes shorter training time than CNH for Bi-CVRP20 and Bi-CVRP50, the time required increases drastically as the problem size grows. As a result, CNH is much more efficiently trained than PMOCO on Bi-CVRP100. Since CNH is trained only once across all sizes, we conclude that it requires less training overload than the baselines, which are trained on each respective problem size.

Instance Augmentation. To further improve the performance of CNH at the inference stage, we apply the instance augmentation proposed in [32], which is also used in PMOCO [17]. In this paper, we interpret a VRP instance as a graph and obtain its augmented views by rotation or flipping transformations, according to the inherent symmetry. For example, if we flip nodes within the square $[0,1] \times [0,1]$ based on $x = 0.5$, the

[2]https://github.com/kevin031060/GeneticLocalSearchTSP
[3]https://github.com/kevin031060/RL_TSP_4static
[4]https://github.com/zhangzizhen/ML-DAM
[5]https://github.com/Xi-L/PMOCO

optimal solution (i.e., the optimal node sequence) to the VRP instance cannot be changed. With the augmented views, i.e., transformed instances, we can solve them in parallel, which means solving the original instance from its different facets. This approach aids in the enhancement of model performance by solving a VRP instance from varied perspectives. Given a coordinate (x,y) in a VRP, there are eight simple transformations, i.e., $(x',y') = (x,y);(y,x);(x,1-y);(y,1-x);(1-x,y);(1-y,x);\ (1-x,1-y);(1-y,1-x)$. In our experiment, we adopt these transformations for each objective, respectively. Hence, we could have 8 transformations for Bi-CVRP (since there is only one coordinate for each node), $8^2=64$ transformations for Bi-TSP and $8^3=512$ transformations for Tri-TSP, respectively. To ensure reasonable inference time, we only use 64 transformations for solving Tri-TSP in CNH and PMOCO. For each instance, we arrange it and its transformations into batches to be tested in parallel. Among all solutions to the transformed instances (and the original instance), the best solution (to the original instance) can be easily found by comparing the objective values of all solutions.

Metrics & **Inference.** The evaluation is based on three metrics (averaged/summed over all testing instances for each problem), including the average hypervolume (HV) [46], average optimality gap (Gap), and the total runtime (Time). HV is a commonly used metric to evaluate both the convergence and diversity of a solution set. We normalize HV values into $[0,1]$ with respect to the same reference point for all methods, and the higher HV the better. The Gap is the ratio of hypervolume difference with respect to the best HV among all methods. We highlight the best HV and Gap by **bold** throughout the paper. We also use "-Aug" to mark the results enhanced by instance augmentation. Since DRL-MOA and ML-DAM possess 101/105 models under 101/105 fixed and uniformly distributed preferences for bi/tri-objective VRPs, we use the same 101/105 preferences in other decomposition-based heuristics (i.e. MOEA/D, PMOCO, and CNH) for a fair comparison.We implement our CNH and all baselines in Python and test them on a single RTX 3060 GPU and an AMD Ryzen 7 5800X CPU. We will make our implementation code publicly available.

B. Comparison Study

We compare CNH with the baselines on Bi-TSP, Bi-CVRP, and Tri-TSP with three problem sizes (i.e. 20, 50, and 100) and generate 100 random test instances for each problem and size. For each problem, while the neural heuristic baselines are trained for each size, we train only one CNH model over the sizes in Λ.

Results are summarized in TABLE II. The reference points for calculating HV are set as follows: For Bi-TSP20, Bi-TSP50, and Bi-TSP100, the reference points are (15, 15), (30, 30), and (60, 60), respectively; For Bi-CVRP20, Bi-CVRP50, and Bi-CVRP100, the reference points are (15, 3), (40, 3), and (60, 3), respectively; For Tri-TSP20, Tri-TSP50, and Tri-TSP100, the reference points are (15, 15, 15), (30, 30, 30), and (60, 60, 60), respectively. We can observe that the performance of conventional heuristics is good on small-sized problems with 20 nodes and two objectives, but deteriorates drastically on problems of larger sizes and with three objectives. Among the neural heuristics, PMOCO and our CNH perform much better than DRL-MOA and ML-DAM. Particularly, without instance augmentation, the original CNH and PMOCO surpass all baselines on most problems and sizes, except on Bi-TSP20 and Bi-CVRP20. The performance of PMOCO and CNH can be further boosted by applying instance augmentation. Notably, compared with the state-of-the-art neural heuristics PMOCO, our CNH consistently shows better performance on all problems of size 50 and 100, with or without instance augmentation. This validates the advantage of our approach, especially considering that only one CNH model is trained (in comparison to other neural heuristics that perform training exclusively on each problem size), and the results on problems of size 50 are actually obtained by zero-shot generalization (since $50 \notin \Lambda$). From a practical view, CNH can make it more convenient to directly produce high-quality solutions to different-sized MOVRP instances, which alleviates the need of retraining.

As for computational efficiency, conventional heuristics cost much longer runtime than neural heuristics, which increases drastically with the problem size. The conventional heuristics rely on iterative search schemes for solving each instance, requiring numerous iterations to enhance solution quality. In contrast, the learning-based methods construct solutions to solve subproblems (i.e. SOVRPs) in parallel, only once. Meanwhile, the end-to-end models are able to efficiently reason out the solutions with pure computation in the neural network, and hence they are more efficient than non-learning heuristic baselines. While CNH and PMOCO share similar runtime, they are relatively slower than DRL-MOA and ML-DAM owing to the sampling of multiple tours for each instance, as stated in Section IV-D. However, DRL-MOA and ML-DAM require massive training overhead for separate models of each subproblem and perform relatively poorly among the neural heuristics. In summary, we conclude that CNH achieves a good trade-off between solution quality and computational efficiency.

C. Study on Instance-Preference Interaction

To validate the instance-preference interaction strategy in our CNH, we develop a similar neural heuristic while excluding the instance-preference interaction, referred to as CNH w/o Interaction. The only difference between CNH and CNH w/o Interaction lies in their encoders. The CNH w/o Interaction encoder consists of two sub-encoders, one for nodes (instance context) and the other for preferences. The former embeds node features using L self-attention layers, while the latter projects preferences into high-dimensional vectors using a Multilayer Perceptron (MLP) with two hidden layers, each with a GELU activation function. The outputs of the sub-encoders are added together to form the final output by the encoder. Unlike CNH, the CNH w/o Interaction does not explicitly involve the interplay between the instance context and preference since they do not pass messages to each other. We compared the performance of CNH and CNH w/o Interaction

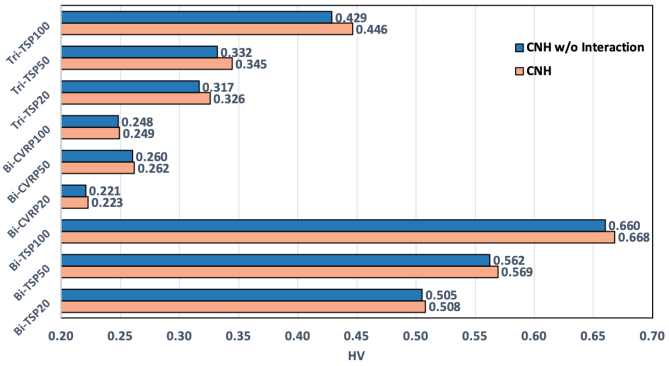


Fig. 2. The efficiency of instance-preference interaction in CNH.

on Bi-TSP, Bi-CVRP, and Tri-TSP with problem sizes of 20, 50, and 100 using the same test instances in Section V-B, and the results are depicted in Fig. 2. Our findings indicate that CNH consistently outperforms CNH w/o Interaction, particularly on Bi-TSP and Tri-TSP, where the superiority is most significant. These results suggest that instance-preference interaction can effectively enhance the performance of neural heuristics by extracting expressive representations of accurate Pareto fronts for MOVRP.

D. Generalization Study

To assess the generalization capability of CNH, we evaluate its zero-shot performance on 100 random test instances for each problem with larger sizes, i.e., 150 and 200, which are unseen during training. We also test neural heuristic baselines on the same instances using their models that are trained for problem size 100 (rather than 20 and 50), so as to gain their best zero-shot performance. All results are gathered in TABLE III. The reference points for calculating HV are set as follows: Regarding Bi-TSP150 and Bi-TSP200, the reference points are (90, 90) and (120, 120), respectively; Regarding Bi-CVRP150 and Bi-CVRP200, the reference points are (120, 3) and (180, 3), respectively; Regarding Tri-TSP150 and Tri-TSP200, the reference points are (90, 90, 90) and (120, 120, 120), respectively. As shown, the original CNH outstrips all baselines for each problem and size. When the instance augmentation is applied, CNH-Aug further achieves the largest HV among all methods across all problems and sizes, with the Gap consistently being 0.00%. Therefore, it is verified that the proposed CNH possesses a favorable generalization capability in solving larger MOVRPs.

E. Benchmark Study

To further verify the effectiveness of CNH, we compare it with the baselines on 11 benchmark instances for Bi-TSP [47], including 10 instances of Bi-TSP100 (i.e. kroAB100, kroAC100, kroAD100, kroBC100, kroBD100, kroCD100, euclAB100, euclCD100, euclEF100, and clusAB100) and 1 instance of Bi-TSP150 (i.e. kroAB150). The kroAB100∼kroCD100 and kroAB150 are generated from kroA100, kroB100, kroC100, kroD100, kroA150 and kroB150 in TSPLIB, according to [47]. We choose the above instances for benchmarking as they are commonly used in the previous neural heuristics [2], [17], [18]. The nodes in euclAB100∼euclEF100 are randomly located from a uniform distribution while those in clusAB100 are randomly clustered in a plane. TABLE IV displays the results, highlighting the best solutions except for the exact ones obtained through exhaustive search [20]. We also include the number of non-dominated solutions ($|NDS|$) for each algorithm in TABLE V, which reflects the diversity of the solution set when the HV values are similar. Intuitively, a larger $|NDS|$ could mean a larger HV only when the solutions of the compared methods are of similar qualities. A larger $|NDS|$ generally indicates better algorithm performance when HV values are close among the methods. According to the results, we observe that CNH achieves a significantly smaller Gap than the baselines without instance augmentation. Additionally,

TABLE IV
RESULTS ON BENCHMARK INSTANCES.

Method	kroAB100 HV	kroAB100 Gap	kroAB100 Time	kroAC100 HV	kroAC100 Gap	kroAC100 Time	kroAD100 HV	kroAD100 Gap	kroAD100 Time
Exact PF	0.782	0.00%	58h	0.783	0.00%	30h	0.785	0.00%	21h
MOEA/D	0.703	10.10%	7.72m	0.693	11.57%	7.72m	0.701	10.76 %	7.72m
NSGA-II	0.696	10.98%	6.52m	0.680	13.15%	6.52m	0.692	11.80%	6.52m
NSGA-III	0.681	12.89%	9.17m	0.655	16.37%	9.08m	0.681	13.27%	9.08m
MOGLS	0.684	12.48%	10.50m	0.687	12.24%	10.50m	0.686	12.66%	10.50m
DRL-MOA	0.560	28.34%	3.95s	0.585	25.32%	3.62s	0.586	25.38%	3.63s
ML-DAM	0.739	5.49%	6.52s	0.746	4.81%	6.23s	0.734	6.54%	6.17s
PMOCO	0.761	2.58%	3.82s	0.764	2.48%	3.46s	0.769	2.08%	3.45s
PMOCO-Aug	0.769	1.56%	13.44s	0.772	1.44%	13.02s	0.774	1.43%	13.04s
CNH	0.767	1.91%	0.58s	0.769	1.84%	0.20s	0.771	1.81%	0.20s
CNH-Aug	**0.773**	**1.05%**	11.90s	**0.775**	**1.10%**	11.48s	**0.776**	**1.11%**	11.48s

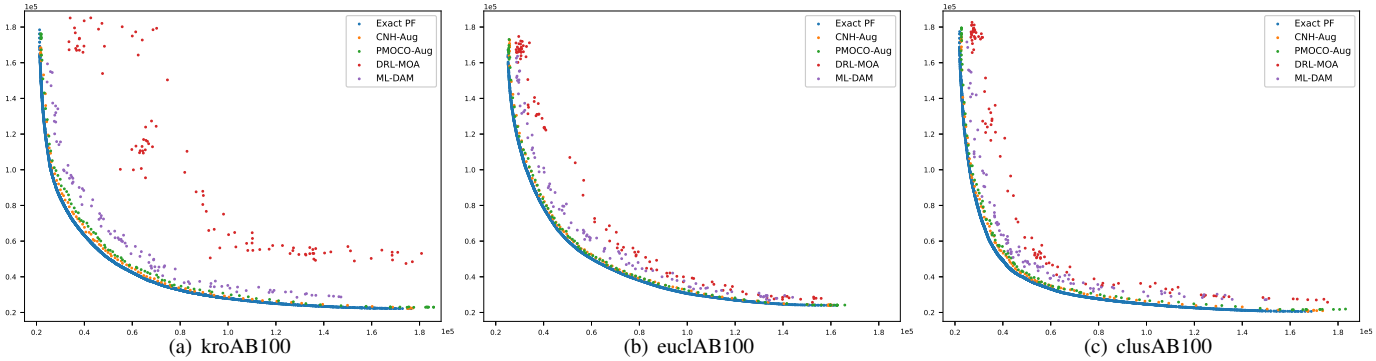
Method	kroBC100 HV	kroBC100 Gap	kroBC100 Time	kroBD100 HV	kroBD100 Gap	kroBD100 Time	kroCD100 HV	kroCD100 Gap	kroCD100 Time
Exact PF	0.784	0.00%	28h	0.781	0.00%	23h	0.791	0.00%	21h
MOEA/D	0.699	10.77%	7.72m	0.703	9.98%	7.72m	0.704	11.02%	7.72m
NSGA-II	0.707	9.85%	6.52m	0.696	10.88%	6.52m	0.704	11.05%	6.52m
NSGA-III	0.675	13.82%	9.10m	0.667	14.61%	9.08m	0.689	12.89%	9.07m
MOGLS	0.687	12.36%	10.58m	0.689	11.84%	10.40m	0.693	12.42%	10.27m
DRL-MOA	0.612	21.88%	3.60s	0.605	22.61%	3.60s	0.586	25.90%	3.60s
ML-DAM	0.733	6.47%	6.28s	0.742	5.03%	6.28s	0.751	5.07%	6.27s
PMOCO	0.766	2.25%	3.47s	0.764	2.21%	3.42s	0.772	2.45%	3.44s
PMOCO-Aug	0.773	1.33%	13.04s	0.770	1.46%	13.03s	0.781	1.33%	13.03s
CNH	0.771	1.63%	0.20s	0.767	1.86%	0.20s	0.780	1.48%	0.20s
CNH-Aug	**0.776**	**0.96%**	11.50s	**0.773**	**1.11%**	11.50s	**0.784**	**0.97%**	11.50s

Method	euclAB100 HV	euclAB100 Gap	euclAB100 Time	euclCD100 HV	euclCD100 Gap	euclCD100 Time	euclEF100 HV	euclEF100 Gap	euclEF100 Time
Exact PF	0.682	0.00%	16h	0.667	0.00%	34h	0.679	0.00%	23h
MOEA/D	0.591	13.39%	10.88m	0.571	14.47%	10.90m	0.580	14.58%	10.92m
NSGA-II	0.587	13.99%	8.90m	0.560	16.01%	8.97m	0.572	15.80%	8.95m
NSGA-III	0.560	17.99%	6.05m	0.548	17.90%	6.10m	0.562	17.23%	6.07m
MOGLS	0.556	18.46%	15.86m	0.538	19.41%	15.56m	0.552	18.74%	15.68m
DRL-MOA	0.606	11.17%	3.99s	0.589	11.69%	3.65s	0.600	11.68%	3.64s
ML-DAM	0.642	5.96%	6.58s	0.627	6.03%	6.32s	0.641	5.52%	6.22s
PMOCO	0.665	2.55%	3.86s	0.650	2.61%	3.48s	0.664	2.25%	3.61s
PMOCO-Aug	**0.672**	**1.47%**	13.48s	0.657	1.56%	13.12s	0.670	1.36%	13.19s
CNH	0.667	2.23%	0.59s	0.652	2.32%	0.21s	0.665	1.99%	0.21s
CNH-Aug	**0.672**	**1.47%**	12.16s	**0.658**	**1.42%**	11.50s	**0.671**	**1.24%**	11.66s

Method	clusAB100 HV	clusAB100 Gap	clusAB100 Time	kroAB150 HV	kroAB150 Gap	kroAB150 Time	Tri-TSP15 HV	Tri-TSP15 Gap	Tri-TSP15 Time
Exact PF	0.780	0.00%	27h	0.728	0.00%	days	0.416	0.00%	5h
MOEA/D	0.707	9.30%	10.78m	0.589	19.12%	10.25m	0.343	17.46%	3.30m
NSGA-II	0.689	11.68%	8.98m	0.532	26.95%	9.32m	0.407	2.14%	1.35m
NSGA-III	0.688	11.74%	9.18m	0.536	26.36%	13.35m	0.395	5.05%	1.67m
MOGLS	0.689	11.62%	11.00m	0.549	24.61%	18.03m	**0.415**	**0.06%**	1.30m
DRL-MOA	0.710	8.99%	3.96s	0.651	10.49%	5.69s	0.395	5.08%	1.18s
ML-DAM	0.740	5.14%	7.28s	0.685	5.94%	9.71s	0.384	7.70%	1.98s
PMOCO	0.758	2.78%	3.74s	0.708	2.78%	5.67s	0.410	1.30%	1.12s
PMOCO-Aug	0.767	1.63%	14.30s	0.714	1.92%	39.04s	0.412	0.84%	1.47s
CNH	0.764	2.01%	0.58s	0.712	2.14%	0.98s	0.412	0.84%	0.38s
CNH-Aug	**0.771**	**1.21%**	12.13s	**0.716**	**1.59%**	36.34s	0.412	0.84%	0.85s

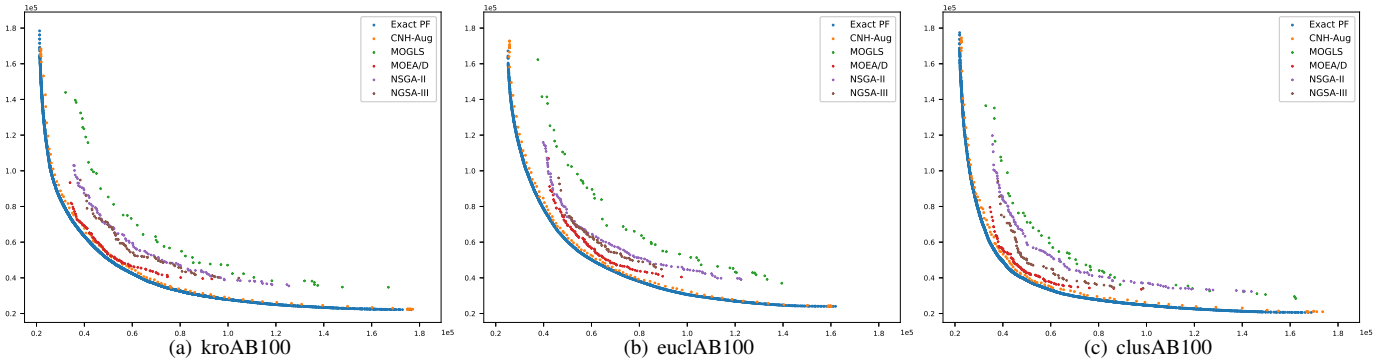
TABLE V
$|NDS|$ ON BENCHMARK INSTANCES.

Method	kroAB100	kroAC100	kroAD100	kroBC100	kroBD100	kroCD100	euclAB100	euclCD100	euclEF100	clusAB100	kroAB150	Tri-TSP15
Exact PF	3332	2458	2351	2752	2657	2044	1812	2268	2530	3036	4701	630
MOEA/D	82	85	80	73	76	66	84	74	75	79	75	18
NSGA-II	100	99	99	100	100	100	100	99	100	99	100	100
NSGA-III	67	58	63	56	54	58	59	62	68	61	46	107
MOGLS	45	37	53	43	42	45	50	61	46	46	68	598
DRL-MOA	17	21	18	17	23	23	40	47	50	38	51	46
ML-DAM	43	41	45	37	38	48	49	49	43	44	58	47
PMOCO	73	69	71	72	69	80	77	77	74	59	79	48
PMOCO-Aug	82	81	81	84	81	82	84	77	81	69	84	41
CNH	68	77	75	71	69	71	73	76	73	64	79	45
CNH-Aug	81	81	81	83	81	75	77	79	75	79	83	44



(a) kroAB100 (b) euclAB100 (c) clusAB100

Fig. 3. Exact/approximated PFs obtained by neural heuristics.



(a) kroAB100 (b) euclAB100 (c) clusAB100

Fig. 4. Exact/approximated PFs obtained by CNH-Aug and conventional heuristics.

CNH-Aug outperforms PMOCO-Aug across all instances except euclAB100, where they perform equally well. Notably, CNH-Aug achieves an impressive result of only around 1% Gap to exact PFs. Among neural heuristics, $|NDS|$ of CNH/CNH-Aug is close to that of PMOCO/PMOCO-Aug but significantly larger than those of the others (i.e. DRL-MOA and ML-DAM).

We also report the results on one Tri-TSP benchmark instance with 15 nodes (Tri-TSP15). Please note that tri-objective VRP benchmarking is rarely investigated in previous neural heuristics, and we conduct this relevant and meaningful comparison to all baselines to further verify the power of CNH. Concretely, the comparative neural heuristics (i.e. DRL-MOA, ML-DAM, and PMOCO) are employed to solve the Tri-TSP15 benchmark instance using models trained on 20-node instances. Since the CNH training process excludes the 15-node instances, the comparison is fair. The results presented in TABLE IV indicate that CNH produces solutions that are closer to exact PF than the other neural heuristics without instance augmentation. According to the results in TABLE V, it is interesting that the $|NDS|$ of CNH is lower than that of MOGLS, which seems to contradict its advantage on bi-TSP instances. However, CNH gains a PF with desirable spread and convergence, so its HV is very close to that of MOGLS and significantly better than the other approximate baselines. Furthermore, CNH shows advantageous computational efficiency and can swiftly generate additional solutions in parallel when provided with more references, which offers more flexibility to practical decision-making.

Visualization. In Fig. 3, we present the exact PFs and approximate PFs obtained by neural heuristics (i.e. DRL-MOA, ML-DAM, PMOCO-Aug, and CNH-Aug) on three representative

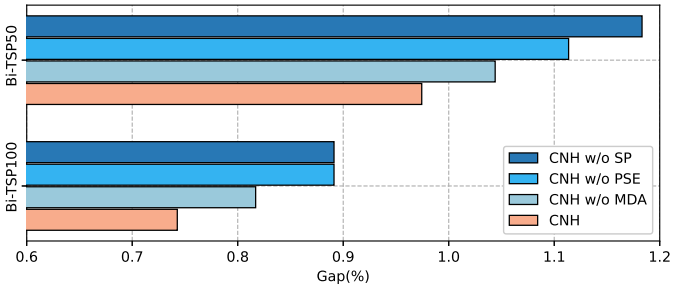


Fig. 5. Ablation study on MDA, PSE, and SP.

TABLE VI
ABLATION STUDY ON ATTENTION MECHANISMS.

Method	MOTSP50			MOTSP100		
	HV	Gap	Time	HV	Gap	Time
CNH	0.569	0.97%	8.34s	0.668	0.74%	24.71s
CNH with MTA	0.568	1.10%	8.33s	0.667	0.88%	23.00s

instances (kroAB100, euclAB100, and clusAB100) that originate from different distributions. For the sake of clarity, we exclude the PFs obtained by CNH and PMOCO, as their results are similar to those obtained by CNH-Aug and PMOCO-Aug, respectively. From Fig. 3, we can see that CNH-Aug attains solutions closer to the exact PFs than PMOCO-Aug on kroAB100 and clusAB100 while achieving nearly coincident solutions with PMOCO on euclAB100. In addition, CNH-Aug visibly outperforms either DRL-MOA or ML-DAM, further verifying the favorable convergence and diversity of our method. We also display a comparison of the exact PFs and approximate PFs obtained by CNH-Aug and four conventional heuristics (i.e. MOEA/D, NSGA-II, NSGA-III, and MOGLS) in Fig. 4. The outcomes demonstrate a noticeable margin between different PFs, with the PFs achieved by CNH-Aug exhibiting as widespread as that of the exact PFs.

Since the node distributions in the benchmark instances differ considerably from the uniform distribution in training, the experimental results also indicate that CNH has a strong out-of-distribution generalization capability.

F. Ablation Study

Components. To analyze the impact of the important components in the proposed method, including multi-head dual-attention (MDA), problem size embedding (PSE) and stochastic preferences (SP), we perform ablation experiments by replacing the MDA with the multi-head self-attention and removing the PSE and SP from CNH separately. We denote the resulting models as CNH w/o MDA, CNH w/o PSE and CNH w/o SP. We compare them with the intact CNH on Bi-TSP50 and Bi-TSP100, and show their Gaps in Fig. 5. We observe that the performance of CNH significantly degrades when any of its components are removed, which reveals the effectiveness of these key designs in the neural architecture or training algorithm.

Attention Mechanism. To further explore the effectiveness of our dual-attention mechanism, we conduct an experiment where we replaced the MDA with a multi-head triple-attention (MTA) mechanism that includes an additional cross-attention from preferences to nodes. The resulting model is referred to as CNH with MTA. The results of CNH and CNH-MTA on Bi-TSP50 and Bi-TSP100 are reported in TABLE VI. Interestingly, the performance of CNH with MTA visibly declines compared to the original CNH. We attribute this to information redundancy caused by the additional attention layer.

VI. LIMITATIONS AND FUTURE WORK

We have identified three limitations of our CNH that require attention and possible solutions for future refinements:

1) **Scalable problem size embedding**. We adopt the sinusoidal positional encoding introduced in the Transformer [38] for embedding the problem size in CNH. However, sinusoidal positional encoding is exploited primarily for sequences of a fixed length, and thus the PSE layer will provide identical problem size information when processing instances exceeding the maximum graph size used in our model, potentially affecting the performance of the CNH. In the future, we aim to investigate scalable problem size embedding techniques to better accommodate large-scale instances.
2) **Extension to the non-Euclidean MOVRPs**. This work is focused on the Euclidean MOVRPs, where distances are calculated using the Euclidean metric. However, some MOVRPs are defined in non-Euclidean spaces, such as those involving real road networks or varied terrain. We plan to extend our research to non-Euclidean MOVRPs in the future.
3) **Adaptive preference selection**. Following the prior works, we currently still use uniformly distributed and predefined preferences to decompose MOVRPs. However, this approach may not be ideal for MOVRPs with irregular Pareto fronts, potentially affecting solving performance. In the future, we will specialize in techniques to efficiently and effectively select preferences for better approximate performance.

VII. CONCLUSION

This paper proposes a novel conditional neural heuristic (CNH) to approximate PF for MOVRPs. We propose a dual-attention-based encoder to well relate preferences and instance contexts, and a size-aware decoder to construct solutions to SOVRPs based on problem size. We train the CNH using a customized REINFORCE algorithm with stochastic preferences. Extensive results show that a single trained CNH can achieve better performance than other neural heuristics trained for each problem size. Moreover, CNH surpasses typical conventional heuristics for MOVRPs, possesses a strong generalization capability, and keeps a relatively small training overhead.

REFERENCES

[1] S. Zajac and S. Huber, "Objectives and methods in multi-objective routing problems: a survey and classification scheme," *European Journal of Operational Research*, vol. 290, no. 1, pp. 1–25, 2021.

[2] Z. Zhang, Z. Wu, H. Zhang, and J. Wang, "Meta-learning-based deep reinforcement learning for multiobjective optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[3] M. Ruchte and J. Grabocka, "Scalable pareto front approximation for deep multi-objective learning," in *2021 IEEE International Conference on Data Mining*. IEEE, 2021, pp. 1306–1311.

[4] M. Ehrgott, *Multicriteria optimization*. Springer Science & Business Media, 2005, vol. 491.

[5] N. Jozefowiez, F. Semet, and E.-G. Talbi, "Multi-objective vehicle routing problems," *European Journal of Operational Research*, vol. 189, no. 2, pp. 293–309, 2008.

[6] J. Kallestad, R. Hasibi, A. Hemmati, and K. Sörensen, "A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems," *European Journal of Operational Research*, 2023.

[7] J. Zhou, Y. Wu, Z. Cao, W. Song, J. Zhang, and Z. Chen, "Learning large neighborhood search for vehicle routing in airport ground handling," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[8] S. M. Raza, M. Sajid, and J. Singh, "Vehicle routing problem using reinforcement learning: Recent advancements," in *Advanced Machine Intelligence and Signal Processing*, 2022, pp. 269–280.

[9] L. Duan, Y. Zhan, H. Hu, Y. Gong, J. Wei, X. Zhang, and Y. Xu, "Efficiently solving the practical vehicle routing problem: A novel joint learning approach," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & data mining*, 2020, pp. 3054–3063.

[10] A. I. Garmendia, J. Ceberio, and A. Mendiburu, "Neural improvement heuristics for graph combinatorial optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[11] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *International Conference on Learning Representations*, 2018.

[12] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems." *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[14] Y. Ma, J. Li, Z. Cao, W. Song, L. Zhang, Z. Chen, and J. Tang, "Learning to iteratively solve routing problems with dual-aspect collaborative transformer," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[15] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[16] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers & Operations Research*, vol. 134, p. 105400, 2021.

[17] X. Lin, Z. Yang, and Q. Zhang, "Pareto set learning for neural multiobjective combinatorial optimization," in *International Conference on Learning Representations*, 2022.

[18] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3103–3114, 2020.

[19] H. Wu, J. Wang, and Z. Zhang, "MODRL/D-AM: Multiobjective deep reinforcement learning algorithm using decomposition and attention model for multiobjective optimization," in *International Symposium on Intelligence Computation and Applications*, 2020, pp. 575–589.

[20] K. Florios and G. Mavrotas, "Generation of the exact pareto set in multi-objective traveling salesman and set covering problems," *Applied Mathematics and Computation*, vol. 237, pp. 1–19, 2014.

[21] L. Paquete and T. Stützle, "Design and analysis of stochastic local search for the multiobjective traveling salesman problem," *Computers & Operations Research*, vol. 36, no. 9, pp. 2619–2631, 2009.

[22] A. Alsheddy and E. E. K. Tsang, "Guided pareto local search based frameworks for biobjective optimization," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[23] C. García-Martínez, O. Cordón, and F. Herrera, "A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP," *European Journal of Operational Research*, vol. 180, no. 1, pp. 116–148, 2007.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[25] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[26] W. Fang, Q. Zhang, J. Sun, and X. Wu, "Mining high quality patterns using multi-objective evolutionary algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3883–3898, 2020.

[27] Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. C. Tan, and Y. Jin, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–34, 2021.

[28] L. Ke, Q. Zhang, and R. Battiti, "A simple yet efficient multiobjective combinatorial optimization method using decompostion and pareto local search," *IEEE Transactions on Cybernetics*, vol. 44, pp. 1808–1820, 2014.

[29] A. Santiago, B. Dorronsoro, A. J. Nebro, J. J. Durillo, O. Castillo, and H. J. Fraire, "A novel multi-objective evolutionary algorithm with fuzzy logic based adaptive selection of operators: FAME," *Information Sciences*, vol. 471, pp. 233–251, 2019.

[30] J.-H. Yi, L.-N. Xing, G.-G. Wang, J. Dong, A. V. Vasilakos, A. H. Alavi, and L. Wang, "Behavior of crossover operators in NSGA-III for large-scale optimization problems," *Information Sciences*, vol. 509, pp. 470–487, 2020.

[31] Y. Xie, S. Yang, D. Wang, J. Qiao, and B. Yin, "Dynamic transfer reference point-oriented MOEA/D involving local objective-space knowledge," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 542–554, 2022.

[32] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min, "Pomo: Policy optimization with multiple optima for reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 21 188–21 198.

[33] Y. Shao, J. C.-W. Lin, G. Srivastava, D. Guo, H. Zhang, H. Yi, and A. Jolfaei, "Multi-objective neural evolutionary algorithm for combinatorial optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[34] Y. Zhang, J. Wang, Z. Zhang, and Y. Zhou, "MODRL/D-EL: Multiobjective deep reinforcement learning with evolutionary learning for multiobjective optimization," in *International Joint Conference on Neural Networks*, 2021, pp. 1–8.

[35] P. Lacomme, C. Prins, and M. Sevaux, "A genetic algorithm for a bi-objective capacitated arc routing problem," *Computers & Operations Research*, vol. 33, no. 12, pp. 3473–3493, 2006.

[36] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[37] E. U. Choo and D. R. Atkins, "Proper efficiency in nonconvex multicriteria programming," *Mathematics of Operations Research*, vol. 8, no. 3, pp. 467–470, 1983.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[40] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[41] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[42] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[43] I. Mehta, S. Taghipour, and S. Saeedi, "Pareto frontier approximation network (PA-Net) to solve bi-objective TSP," in *IEEE 18th International Conference on Automation Science and Engineering*, 2022, pp. 1198–1205.

[44] J. Blank, K. Deb, and P. C. Roy, "Investigating the normalization procedure of nsga-iii," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2019, pp. 229–240.

[45] H. Ishibuchi and T. Murata, "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)*, vol. 28, no. 3, pp. 392–403, 1998.

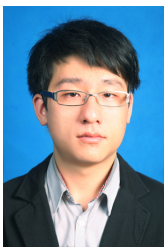
[46] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006.

[47] T. Lust and J. Teghem, "Two-phase Pareto local search for the biobjective traveling salesman problem," *Journal of Heuristics*, vol. 16, no. 3, pp. 475–510, 2010.

Mingfeng Fan received the B.Eng degree in Transport Equipment and Control Engineering from Central South University, Changsha, China, in 2019, where she is currently pursuing the Ph.D. degree in Traffic and Transportation Engineering. Her research interests include machine learning and UAV path planning.

Yaoxin Wu received the B.Eng degree in traffic engineering from Wuyi University, Jiangmen, China, in 2015, the M.Eng degree in control engineering from Guangdong University of Technology, Guangzhou, China, in 2018, and the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2023. He was a Research Associate with the Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU). He joins the Department of Information Systems, Faculty of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, as an Assistant Professor. His research interests include combinatorial optimization, integer programming and deep learning.

Zhiguang Cao received the Ph.D. degree from Interdisciplinary Graduate School, Nanyang Technological University. He received the B.Eng. degree in Automation from Guangdong University of Technology, Guangzhou, China, and the M.Sc. in Signal Processing from Nanyang Technological University, Singapore, respectively. He was a Research Fellow with the Energy Research Institute @ NTU (ERI@N), a Research Assistant Professor with the Department of Industrial Systems Engineering and Management, National University of Singapore, and a Scientist with the Agency for Science Technology and Research (A*STAR), Singapore. He joins the School of Computing and Information Systems, Singapore Management University, as an Assistant Professor. His research interests focus on learning to optimize (L2Opt).



Wen Song received the B.S. degree in automation and the M.S. degree in control science and engineering from Shandong University, Jinan, China, in 2011 and 2014, respectively, and the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2018. He was a Research Fellow with the Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU). He is currently an Associate Research Fellow with the Institute of Marine Science and Technology, Shandong University. His current research interests include artificial intelligence, planning and scheduling, multi-agent systems, and operations research.

Guillaume Sartoretti joined the Mechanical Engineering department at the National University of Singapore as an Assistant Professor in 2019. Before that, he was a Postdoctoral Fellow in the Robotics Institute at Carnegie Mellon University. He received his Ph.D. degree in robotics from EPFL in 2016. He also holds a B.S. and an M.S. degree in Mathematics and Computer Science from the University of Geneva. He is interested in the emergence of collaboration/cooperation in large groups of intelligent agents making individual choices based on their local understanding of the world.

Huan Liu received the B.S. degree in Transportation Engineering from the Wuhan University of Technology, China, in 2018, and the M.S. degree in Traffic and Transportation Engineering from Central South University, Changsha, China, in 2021. She is currently pursuing the Ph.D. degree in traffic and transportation engineering at Central South University, Changsha, China. Her research interests include UAV path planning and task assignment.

Guohua Wu received the B.S. degree in Information Systems and Ph.D degree in Operations Research from National University of Defense Technology, China, in 2008 and 2014, respectively. During 2012 and 2014, he was a visiting Ph.D student at University of Alberta, Edmonton, Canada. He is currently a Professor at the School of Traffic and Transportation Engineering, Central South University, Changsha, China. His current research interests include Planning and Scheduling, Computational Intelligence and Machine Learning. He has authored more than 100 referred papers including those published in IEEE TCYB, IEEE TSMCA and IEEE TEVC. He serves as an Associate Editor of Information Sciences, and an Associate Editor of Swarm and Evolutionary Computation Journal, an editorial board member of International Journal of Bio-Inspired Computation, and Guest Editors of several journals.