Preference-Conditioned Dynamic Attention Model for Multi-objective Capacitated Arc Routing Problem

Xiaoyuan Wei[†], Yang Wang[†*], Ya-Hui Jia[†*], Feng-Feng Wei[‡], Qiang Yang[§], Wei-Neng Chen[‡]
[†]School of Future Technology, South China University of Technology, Guangzhou, China
202421064123@mail.scut.edu.cn, ftwangyang@mail.scut.edu.cn, jia.yahui@foxmail.com
[‡]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
fengfengwei@scut.edu.cn, cschenwn@scut.edu.cn
[§]School of Artificial Intelligence, Nanjing University of Information Science and Technology, Nanjing, China
mmmyq@126.com

***Abstract*—Multi-objective Capacitated Arc Routing Problem (MOCARP) is a complex multi-objective combinatorial optimization problem. Existing Neural Combinatorial Optimization (NCO) methods often treat preference vectors merely as static context, failing to evaluate how dynamic state transitions affect conflicting objectives. To address this, we propose a novel single-model NCO method called the Preference-Conditioned Dynamic Attention Model (PCDAM) to solve MOCARP. First, we introduce a preference-aware attention mechanism that injects preference weights directly into the decoder's query vectors, enabling the model to adaptively adjust decision priorities according to given preferences with minimal computational overhead. Second, we design dynamic objective-aware features by combining objective-specific metrics with preference vectors, enabling the model to evaluate the impact of each action on multiple conflicting objectives in real-time during decoding, which provides a correction to the attention model. Extensive experiments demonstrate that PCDAM significantly outperforms representative evolutionary algorithms and existing NCO methods in terms of hypervolume. Our code is available at https://github.com/huhuGood6/PCDAM.**

***Index Terms*—Capacitated arc routing problems, Multi-objective optimization, Deep reinforcement learning.**

I. Introduction

The Capacitated Arc Routing Problem (CARP) is a classical combinatorial optimization problem [1], [2]. The objective is to find service routes over required edges in a connected graph while satisfying vehicle capacity constraints. Many real-world applications, such as urban waste collection and road maintenance, can be naturally formulated as CARPs [3], [4]. Traditional studies typically consider CARP as a single-objective problem, focusing on minimizing total service cost [5]. However, practical scenarios often involve multiple conflicting objectives [6], [7]. For instance, one may simultaneously minimize overall cost and reduce the maximum workload among vehicles to ensure service balance. In such settings, a unique global optimum no longer exists; instead, the goal becomes generating a set of Pareto-optimal solutions that capture objective trade-offs and accommodate diverse preferences. This gives rise to the Multi-objective Capacitated Arc Routing Problem (MOCARP).

Classical multi-objective algorithms explore these trade-offs using evolutionary operators, dominance relations, or scalarization-based decomposition [8]–[11]. Although effective, they rely on problem-specific expert knowledge and incur substantial computational overhead. Moreover, they generate solutions instance by instance, limiting scalability in large-scale environments.

Recent advances in Neural Combinatorial Optimization (NCO) have achieved remarkable success in Combinatorial Optimization Problems (COPs), such as the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP) [12]. Existing NCO methods for Multi-objective Combinatorial Optimization Problems (MOCOPs) follow two paradigms. 1) Multi-model methods decompose a multi-objective task into scalarized subproblems and train [13], [14] or finetune [15], [16] a separate network for each subproblem. These methods become prohibitively expensive as the preference space grows. 2) Single-model methods incorporate preference vectors as conditional inputs, enabling one unified model to handle all subproblems [17]. Prior work includes hypernetwork-based methods that adjust decoder parameters [17] and encoder-level mechanisms that fuse preference information into node embeddings [18]. These methods target node routing problems (e.g., TSP, VRP), while CARP involves arc routing with fundamentally different graph structures. To the best of our knowledge, no NCO method has been developed for MOCARP. Moreover, directly extending existing methods faces two key limitations. First, they treat preference vectors as static context with limited coupling to the attention-based decoding process, hindering adaptive adjustment of decision priorities. Second, they rely on static graph embeddings and overlook dynamic state features, such as accumulated costs, that reflect real-time objective evolution, limiting the capacity to evaluate how each action affects multiple objectives dynamically.

To address these challenges, we propose a Preference-Conditioned Dynamic Attention Model (PCDAM), a novel single-model NCO framework tailored to MOCARP. The core idea is to deeply integrate preference information and dynamic state features within the attention-based decoding

*Corresponding author.
This work is supported in part by the National Natural Science Foundation of China under Grant 62576172, in part by the introduced innovative R&D team of Guangdong under Grant 2023ZT10L145, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515012028.

mechanism, enabling adaptive decision-making that responds to both given preferences and real-time objective trade-offs. Our main contributions are as follows:

- To the best of our knowledge, we present the first application of NCO to MOCARP, providing a new perspective for solving this problem.
- We introduce a preference-aware attention mechanism that injects preference weights directly into the decoder's query vectors, enabling adaptive adjustment of decision priorities with minimal computational overhead.
- We design dynamic objective-aware features integrated with preference vectors, allowing the model to evaluate how each action affects multiple objectives in real-time.

II. Related Work

A. Traditional Methods for MOCOPs

Traditional methods for MOCOPs include exact and metaheuristic methods. Exact algorithms [19] aim to identify the full Pareto set but become impractical for NP-hard problems due to exponential complexity. Metaheuristic methods, particularly Multi-objective Evolutionary Algorithms (MOEAs), thus have been widely adopted. Deb *et al.* [8] and Zhang *et al.* [9] established mainstream paradigms based on dominance (NSGA-II) and decomposition (MOEA/D), respectively, with subsequent enhancements such as adaptive weight adjustment [20]. However, these methods rely heavily on handcrafted operators and extensive iterative search, resulting in high computational costs.

B. NCO Methods for COPs

Distinguished from traditional methods, NCO employs end-to-end deep neural networks to learn solution strategies directly from problem instances [1], [12], [21]. A seminal breakthrough was established by Kool *et al.* [22] with the Attention Model (AM), which adapted the Transformer architecture to routing problems and demonstrated superior modeling capabilities. Building on this, Kwon *et al.* [23] introduced Policy Optimization with Multiple Optima (POMO), leveraging solution space symmetries to enhance training stability and performance. POMO has since emerged as a widely recognized baseline in COPs studies. Overall, the evolution from sequence generation models to attention-based architectures has established a novel paradigm for combinatorial optimization.

C. NCO Methods for MOCOPs

NCO methods for MOCOPs predominantly employ decomposition strategies under multi-model and single-model paradigms. The multi-model paradigm trains a dedicated network for each subproblem [13], [14], or employs meta-learning to facilitate fast adaptation [15], [16]. However, this paradigm incurs substantial computational overhead. In contrast, the single-model paradigm [17], [18], [24]–[27] has garnered significant attention for its superior efficiency. The core challenge in this paradigm lies in the effective injection of preference information. Existing methods can be broadly categorized into two streams based on their injection mechanisms: parameter-space injection and feature-space injection. The former, represented by PMOCO [17], utilizes a hyper-network to dynamically generate decoder parameters conditioned on the preference vector. The latter, including WE-CA [18] and CNH [26], integrates preference information directly into the embedding space or context vectors to guide the decoding process.

Although the methods mentioned above have achieved significant success in node routing problems, their application to MOCARP remains to be explored. Moreover, existing single-model architectures exhibit two limitations when extended to MOCARP. First, static preference injection lacks coupling to the attention-based decoding process, hindering adaptive adjustment of decision priorities. Second, reliance on static embeddings overlooks dynamic state features that reflect real-time objective evolution, thereby limiting state-aware trade-offs essential for high-quality Pareto front. Therefore, this paper proposes a novel single-model NCO method that integrates preference information into the attention mechanism and incorporates dynamic state features for adaptive decision-making in MOCARP.

III. Preliminaries

A. MOCARP Formulation

A MOCARP instance is defined on an undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and required edges (tasks), respectively. A depot is located at node $v_d \in \mathcal{V}$. Each edge $e \in \mathcal{E}$ is associated with a positive demand $q_e > 0$ and a non-negative traversal cost $c_e \geq 0$, requiring service. The fleet consists of m homogeneous vehicles, each possessing a capacity of $L_{\max}$, all of which originate from the depot.

A solution to MOCARP is a set of routes $\mathbf{\Pi} = \{s_1, s_2, \ldots, s_m\}$. Each route $s_i (1 \leq i \leq m)$ starts and ends at depot v_d and consists of an ordered sequence of required edges $(s_i = (e_{(i,1)}, e_{(i,2)}, \ldots, e_{(i,\mathcal{N}_i)}))$, where $\mathcal{N}_i$ is the number of required edges in route s_i, and $e_{(i,j)} \in \mathcal{E}$ denotes the j-th required edge serviced by vehicle i. A solution $\mathbf{\Pi}$ is feasible if: 1) every task is serviced exactly once; 2) the total demand of each route does not exceed $L_{\max}$; and 3) route continuity is maintained. The MOCARP with M objectives is formally defined as:

$$\min_{\mathbf{\Pi} \in \mathcal{X}} \quad \boldsymbol{F}(\mathbf{\Pi}) = (f_1(\mathbf{\Pi}), f_2(\mathbf{\Pi}), \ldots, f_M(\mathbf{\Pi})), \tag{1}$$

where $\mathcal{X}$ denotes the feasible solution space, and $\boldsymbol{F}(\mathbf{\Pi})$ is an M-dimensional objective vector. In this study, we consider $M = 2$ conflicting objectives: the total traversal cost (f_1) and the makespan (f_2). Since these objectives conflict, solutions are evaluated based on Pareto optimality.

Pareto Dominance. A solution $\mathbf{\Pi}^A \in \mathcal{X}$ dominates another solution $\mathbf{\Pi}^B \in \mathcal{X}$ (denoted as $\mathbf{\Pi}^A \prec \mathbf{\Pi}^B$) if $f_i(\mathbf{\Pi}^A) \leq f_i(\mathbf{\Pi}^B)$ for all $i \in \{1, \ldots, M\}$, and $f_j(\mathbf{\Pi}^A) < f_j(\mathbf{\Pi}^B)$ for at least one $j \in \{1, \ldots, M\}$.

Pareto Optimality. A solution $\mathbf{\Pi}^* \in \mathcal{X}$ is Pareto optimal if no other feasible solution dominates it. The set of all Pareto optimal solutions forms the Pareto Set ($\mathcal{P}$), and its image in

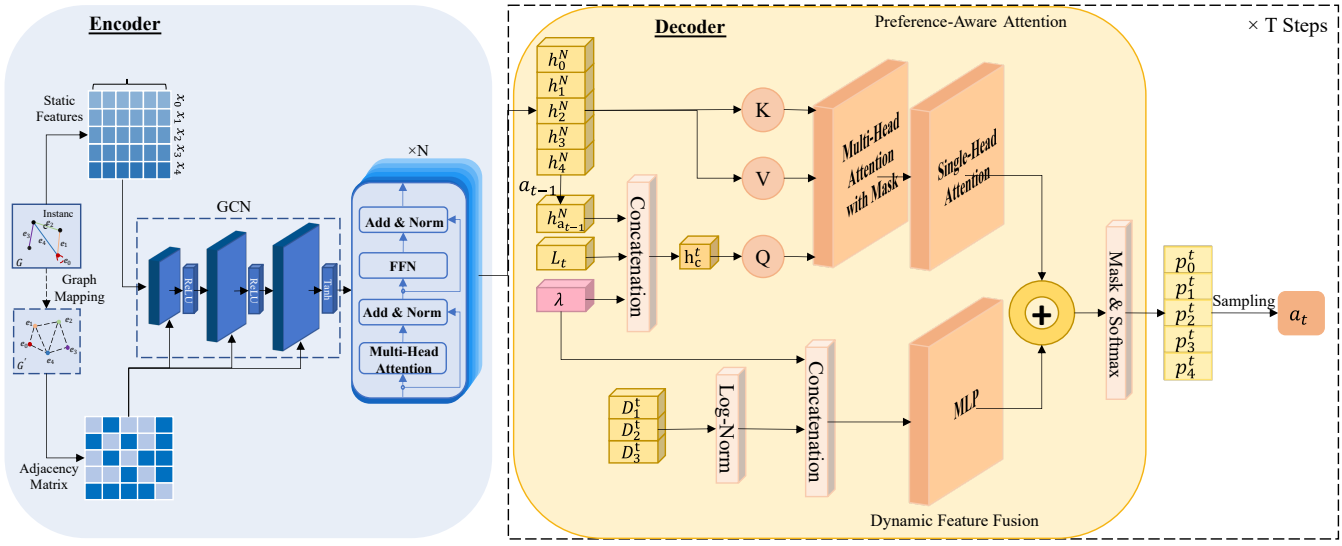


Fig. 1: The Architecture of PCDAM. The encoder extracts structural information from static features via GCN, followed by N stacked layers (MHA and FFNs) to construct enriched representations. The dynamic attention decoder comprises two components: 1) a preference-aware attention module that injects preference $\boldsymbol{\lambda}$ into query vectors to guide topological reasoning, and 2) a dynamic objective-aware feature module that computes correction biases from real-time state features (e.g., accumulated costs). The two components are fused to generate action probabilities.

the objective space, $\mathcal{F} = \{\boldsymbol{F}(\boldsymbol{\Pi}) \mid \boldsymbol{\Pi} \in \mathcal{P}\}$, is the Pareto front.

B. Decomposition-based Methods

We employ decomposition strategies [9] to transform the multi-objective problem into scalar subproblems, each defined by a weight vector $\boldsymbol{\lambda} \in \mathbb{R}^M$ satisfying $\sum_{i=1}^{M} \lambda_i = 1$ and $\lambda_i \geq 0$, representing a preference. This formulation allows the DRL agent to learn a policy $\pi_\theta(\boldsymbol{\Pi} \mid \mathcal{G}, \boldsymbol{\lambda})$ that generates a solution $\boldsymbol{\Pi}$ for a given instance $\mathcal{G}$ and preference $\boldsymbol{\lambda}$. The reward is defined as the negative scalarized objective: $R(\boldsymbol{\Pi}|\boldsymbol{\lambda}) = -g(\boldsymbol{\Pi} \mid \boldsymbol{\lambda})$. Two scalarization functions are defined as follows:

Weighted Sum (WS). WS linearly combines objectives: $g_{\text{ws}}(\boldsymbol{\Pi} \mid \boldsymbol{\lambda}) = \sum_{i=1}^{M} \lambda_i f_i(\boldsymbol{\Pi})$. While efficient, WS is limited to finding solutions on convex regions of the Pareto front.

Tchebycheff (TCH). To handle non-convex fronts, Tch minimizes the maximum weighted deviation from an ideal point $\boldsymbol{z}^*$:

$$g_{\text{tch}}(\boldsymbol{\Pi} \mid \boldsymbol{\lambda}, \boldsymbol{z}^*) = \max_{i \in \{1,\dots,M\}} \{\lambda_i \, |f_i(\boldsymbol{\Pi}) - z_i^*|\}\,. \tag{2}$$

By varying $\boldsymbol{\lambda}$, the agent learns to approximate the entire Pareto front.

IV. Methodology

A. The Model Structure

The proposed PCDAM is an end-to-end neural framework tailored for MOCARP. As illustrated in Fig. 1, PCDAM adopts an encoder-decoder architecture [22]. It features an encoder comprising a Graph Convolutional Network (GCN) and stacked attention layers to capture graph topology, and a dynamic attention decoder. The decoder explicitly integrates the preference vector $\boldsymbol{\lambda}$ into the decision process via two components: 1) a preference-aware attention mechanism that injects preferences into query vectors to guide topological focus, and 2) dynamic objective-aware features that fuse preferences with real-time state information to generate correction biases. This design enables the model to efficiently approximate the Pareto front by generating solutions highly responsive to varying objective weights.

B. Encoder

Our encoder adopts a Transformer-based architecture [22]. Since the MOCARP instance is defined on an edge-weighted graph, we employ a graph transformation strategy [28] to convert the original edge-weighted graph into a node-weighted graph. Specifically, each service edge e_i in the original graph is mapped to a node in the new task graph. This transformation allows us to explicitly define the topological connectivity between tasks, facilitating the construction of the normalized adjacency matrix $\hat{\mathbf{A}}$ required for the GCN.

For an instance containing $\mathcal{M}$ service edges, edge i is defined by a 6-dimensional static feature vector $\mathbf{x}_i$:

$$\mathbf{x}_i = [I(u), I(v), D(u, v_d), D(v, v_d), \tilde{\tau}(e_i), \tilde{q}(e_i)], \tag{3}$$

where $I(\cdot)$ is the depot indicator function, $D(\cdot, v_d)$ denotes the shortest path distance to the depot, $\tilde{\tau}(e_i)$ is the normalized deadheading cost, and $\tilde{q}(e_i)$ is the normalized demand.

The feature vectors of all $\mathcal{M}$ tasks are stacked to form the initial edge feature matrix $\mathbf{X}^{(0)}$, where the i-th row corresponds

to $\mathbf{x}_i$. To fuse these static attributes with the graph topology, we employ a three-layer GCN for initial embedding:

$$\mathbf{X}^{(r+1)} = \sigma(\hat{\mathbf{A}}\mathbf{X}^{(r)}\mathbf{W}^{(r)}), \tag{4}$$

where $\mathbf{X}^{(r)}$ denotes the edge embedding matrix at layer r ($r \in \{0, 1, 2\}$), $\mathbf{W}^{(r)}$ is the learnable weight matrix, and $\sigma(\cdot)$ is a non-linear activation function.

The GCN output $\mathbf{X}^{(3)}$ serves as the initial input $\mathbf{H}^{(0)}$ for the N stacked attention layers ($N = 6$ in this study). For each layer $l \in \{1, \ldots, N\}$, we update the feature vector $\mathbf{h}_i^{(l)}$ for each edge i using Multi-Head Attention (MHA) and Feed-Forward (FF) sublayers, both followed by skip-connection and Instance Normalization (IN):

$$\hat{\mathbf{h}}_i^{(l)} = \text{IN}\left(\mathbf{h}_i^{(l-1)} + \text{MHA}\left(\mathbf{h}_i^{(l-1)}, \mathbf{H}^{(l-1)}\right)\right), \tag{5}$$

$$\mathbf{h}_i^{(l)} = \text{IN}\left(\hat{\mathbf{h}}_i^{(l)} + \text{FF}\left(\hat{\mathbf{h}}_i^{(l)}\right)\right). \tag{6}$$

After N layers, we obtain the final edge embeddings $\mathbf{H}^{(N)} = \{\mathbf{h}_1^{(N)}, \mathbf{h}_2^{(N)}, \ldots, \mathbf{h}_{\mathcal{M}}^{(N)}\}$, which serve as the context for decoding.

C. Dynamic Attention Decoder

The decoder computes edge selection probabilities at each step t in an autoregressive manner. To address the limited responsiveness to preferences in existing methods, we propose two complementary components: preference-aware attention and dynamic objective-aware features.

1) Preference-Aware Attention: This component injects preference information directly into the attention mechanism to guide topological reasoning. The static edge embeddings $\mathbf{H}^{(N)}$ are projected into keys and values, which remain fixed throughout decoding:

$$\mathbf{K} = \mathbf{W}^K\mathbf{H}^{(N)}, \quad \mathbf{V} = \mathbf{W}^V\mathbf{H}^{(N)}. \tag{7}$$

At step t, we construct a preference-aware context vector $\mathbf{h}_c^t$ by concatenating the embedding of the last selected edge a_{t-1}, the current vehicle's remaining capacity L_t, and the preference vector $\boldsymbol{\lambda}$:

$$\mathbf{h}_c^t = [\mathbf{h}_{a_{t-1}}^N \parallel L_t \parallel \boldsymbol{\lambda}]. \tag{8}$$

where $[\cdot||\cdot]$ denotes concatenation along the feature dimension. A query vector is then projected from $\mathbf{h}_c^t$:

$$\mathbf{Q}_c^t = \mathbf{W}^Q\mathbf{h}_c^t. \tag{9}$$

By explicitly including $\boldsymbol{\lambda}$ in the query, the attention distribution dynamically adjusts based on the current trade-off priority. We apply MHA with mask to refine the context, followed by single-head attention to compute base compatibility scores:

$$\mathbf{H}_c^{'} = \text{MaskedMHA}(\mathbf{Q}_c^t, \mathbf{K}, \mathbf{V}, \mathbf{M}^t), \tag{10}$$

$$\mathbf{S}_1^t = C \cdot \tanh\left(\frac{(\mathbf{W}_s^Q\mathbf{H}_c^{'})(\mathbf{W}_s^K\mathbf{H}^{(N)})^\top}{\sqrt{d_k}}\right), \tag{11}$$

where $\mathbf{M}^t$ masks infeasible edges (visited or exceeding capacity), $C = 10$ is a clipping constant, and d_k is the key dimension.

2) Dynamic Objective-Aware Features: This component computes correction biases from real-time state features that reflect objective evolution during decoding. We define three features:

- **Marginal Cost ($D_{1,i}^t$):** The traversal cost from the current location to candidate task i.
- **Accumulated Cost (D_2^t):** The total cost incurred by the current vehicle up to step t.
- **Current Makespan (D_3^t):** The maximum duration among all routes constructed so far.

These features are log-normalized ($\hat{D} = \log(1 + |D|)$) to stabilize training. For each candidate edge i, we concatenate its marginal cost with global states and the preference vector $\boldsymbol{\lambda}$, then pass through a Multi-Layer Perceptron (MLP):

$$\mathbf{S}_2^t = \text{MLP}([D_1^t \parallel D_2^t \parallel D_3^t \parallel \boldsymbol{\lambda}]). \tag{12}$$

This bias $\mathbf{S}_2^t$ acts as a correction term that encourages or penalizes candidates based on alignment with $\boldsymbol{\lambda}$ given the current state. For example, if the makespan is high (the lower the better) and its preference weight is large, the network learns to penalize candidates that further increase the longest route.

3) Policy Generation: The final logit u_i^t is obtained by fusing the attention score $S_{1,i}^t$ and the dynamic bias $S_{2,i}^t$:

$$u_i^t = \begin{cases} S_{1,i}^t + S_{2,i}^t, & \text{if edge } i \text{ is feasible} \\ -\infty, & \text{otherwise} \end{cases}. \tag{13}$$

The logits are transformed into selection probabilities via Softmax:

$$p_\theta(a_t = i \mid \mathcal{G}, \boldsymbol{\lambda}, a_{0:t-1}) = \frac{\exp(u_i^t)}{\sum_j \exp(u_j^t)}. \tag{14}$$

Thereby, the probability of generating a complete solution $\boldsymbol{\Pi}$ for instance $\mathcal{G}$ can be expressed as the product of the conditional probabilities at each time step:

$$p_\theta(\boldsymbol{\Pi} \mid \mathcal{G}, \boldsymbol{\lambda}) = \prod_{t=1}^{T} p_\theta(a_t \mid \mathcal{G}, \boldsymbol{\lambda}, a_{0:t-1}), \tag{15}$$

where T denotes the length of the solution sequence.

D. Training

We leverage the POMO framework to sample $\mathcal{M}$ trajectories with distinct starting edges, generating diverse solutions for reinforcement learning against decomposed scalarized objectives, as shown in Algorithm 1.

We define the reward $R(\boldsymbol{\Pi}|\boldsymbol{\lambda})$ as the negative scalarized objective of solution $\boldsymbol{\Pi}$ under preference $\boldsymbol{\lambda}$. Following REINFORCE, the policy gradient is:

$$\nabla_\theta J(\theta) \approx \frac{1}{\mathcal{M}} \sum_{j=1}^{\mathcal{M}} (R(\boldsymbol{\Pi}_j|\boldsymbol{\lambda}) - b) \nabla_\theta \log p_\theta(\boldsymbol{\Pi}_j \mid \mathcal{G}, \boldsymbol{\lambda}), \tag{16}$$

where the baseline b is computed as the average reward across all rollouts within the current batch:

$$b = \frac{1}{\mathcal{M}} \sum_{j=1}^{\mathcal{M}} R(\boldsymbol{\Pi}_j|\boldsymbol{\lambda}). \tag{17}$$

Algorithm 1 Training of PCDAM

Input: Preference vector $\boldsymbol{\lambda}$, batch size $\mathcal{B}$, number of required edges $\mathcal{M}$, instance set $\mathcal{D}$, epochs $\mathcal{H}$, learning rate β
Output: Optimized parameters θ
Initialize θ
for epoch $= 1$ **to** $\mathcal{H}$ **do**
  for $i = 1$ **to** $\mathcal{B}$ **do**
    $\mathcal{G}_i \leftarrow$ SampleInstance$(\mathcal{D})$
    $\{e_i^1, e_i^2, \ldots, e_i^{\mathcal{M}}\} \leftarrow$ MultiStart$(\mathcal{G}_i)$
    for $j = 1$ **to** $\mathcal{M}$ **do**
      $\Pi_i^j \leftarrow$ SampleRollout$(e_i^j, \mathcal{G}_i, \boldsymbol{\lambda}, \theta)$
    end for
    Compute baseline b by Eq. (17)
    Compute gradient $\nabla_\theta J(\theta)$ by Eq. (16)
    $\theta \leftarrow \theta + \beta \nabla_\theta J(\theta)$
  end for
end for
return θ

V. Experiments

A. Experimental Settings

*1) **Problem and Data Generation**:* We evaluate the proposed PCDAM on three problem scales based on the number of required edges, specifically set to $|\mathcal{E}| \in \{20, 50, 100\}$, hereafter referred to as E20, E50, and E100, respectively. For each instance, edge costs and demands are sampled from a uniform distribution $U[1, 10]$, and the depot is randomly selected. For each scale, we generate 128,000 instances per epoch for training and 200 separate instances for testing.

*2) **Hyperparameters** :* The model is trained for 200 epochs with a batch size of 64 using the Adam optimizer (learning rate 10^{-4}, weight decay 10^{-6}). Following the decomposition strategy described in Section III, we generate 101 weight vectors using the method proposed by Das and Dennis [29], uniformly sampling preferences on the simplex $\sum_{i=1}^{2} \lambda_i = 1$. The hardware setup includes an Intel i9-14900HX CPU for traditional methods and a single RTX 4090D GPU for NCO methods.

*3) **Baselines** :* We compare PCDAM against three categories of strong baselines. For a fair comparison, all NCO baselines were adapted to the MOCARP constraints and retrained from scratch on our generated datasets using the same training budget (200 epochs).

- **Traditional Methods:** The representative MOEAs, NSGA-II [8], is adopted with a population size of 400. It is executed for 3,000 generations, augmented with a local search operator to enhance solution quality.
- **Single-model NCO Methods:** Including PMOCO [17], WE-CA [18], and CNH [26]. All these methods utilize the TCH decomposition strategy defined in Eq. (2), consistent with our setting. Notably, CNH retains its specific perception module for size generalization.
- **Multi-model NCO Methods:** EMNH [15] is included as the state-of-the-art multi-model NCO method driven by meta-learning. We strictly follow its original training protocol, including the pre-training of the meta-model and the fine-tuning steps for specific subproblems.

*4) **Metrics**:* We employ Hypervolume (HV) to evaluate the convergence and diversity of the Pareto front. A higher HV indicates better performance. In addition to HV, we report the Optimality gap (Gap), which measures the relative difference between the HV of a specific method and the best HV obtained among all methods for the same instance. We report the average HV, Gap, and total solving time on the 200 test instances.

B. Performance Comparison

1) Results on Synthetic Datasets: The comparison results on MOCARP instances across three scales (E20, E50, and E100) are reported in Table I. Our proposed PCDAM significantly outperforms the compared baselines in terms of solution quality. Compared with WE-CA, the strongest baseline among single-model neural methods, PCDAM achieves substantially smaller gap across all scales, e.g., 0.00% vs 1.79% on E50 and 0.00% vs 3.12% on E100. While baselines like NSGA-II suffer from severe performance degradation as the problem scale grows (reaching a 57.96% gap on E100), PCDAM maintains robust optimality, achieving a 0.00% gap across all tested sizes. This demonstrates that PCDAM consistently achieves the best approximation quality among the compared methods.

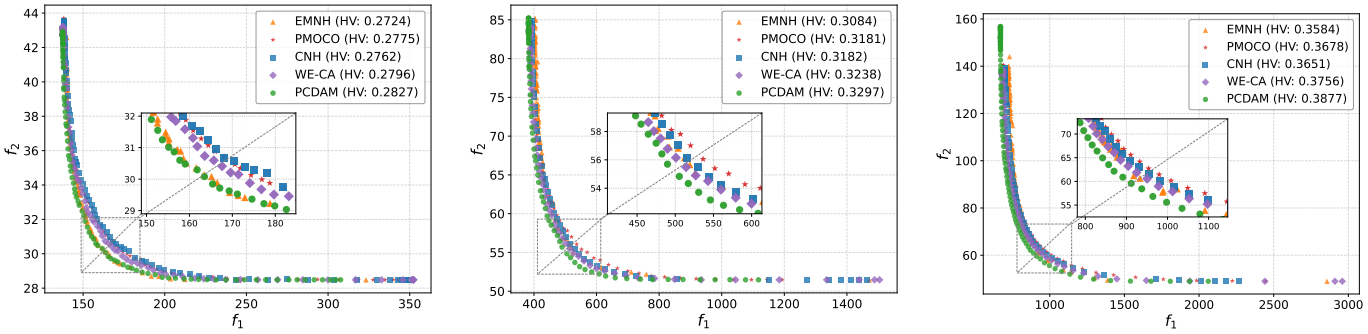
In terms of computational efficiency, compared with the traditional NSGA-II that consumes hours due to iterative search (e.g., 11.52h on E100), our PCDAM only takes 160 seconds. Although PCDAM requires slightly more inference time than pure attention-based models (e.g., PMOCO and WE-CA) due to the computation of dynamic objective-aware features, this overhead is well-justified by the significant improvement in solution quality.

To intuitively assess the solution quality, we visualize the Pareto fronts obtained by different NCO methods across test datasets of various scales in Fig. 2, where the green curve denotes our PCDAM. Specifically, we treat the set of bi-objective values derived from solving the decomposed scalarized subproblems as the approximate Pareto front for plotting. The Pareto fronts produced by PCDAM are consistently closer to the ideal point and exhibit smoother distributions than those of the baselines across all scales. This visually corroborates the 0.00% gap reported in Table I, providing further evidence that PCDAM generates superior approximations of the Pareto front through effective preference conditioning and dynamic state awareness.

2) Results on the gdb Benchmark Dataset: To assess the zero-shot generalization capability of PCDAM, we extend our evaluation to the well-known gdb benchmark [30] using models pretrained on synthetic datasets (E20 and E50). As detailed in Table II, PCDAM exhibits exceptional transferability, achieving the highest mean Hypervolume of 0.4528 and a 0.00% gap. This indicates that PCDAM delivers the best results among the compared algorithms in terms of HV and Gap, significantly surpassing the runner-up EMNH (3.09% gap). Specifically, PCDAM achieves the best solutions on 13 out of 23 instances,

TABLE I: Comparison between PCDAM and Other Algorithms on the Synthetics Dataset.

Method	E20			E50			E100		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
NSGA-II	0.2747	2.83%	1.38h	0.2341	29.00%	3.31h	0.1630	57.96%	11.52h
EMNH	0.2724	3.64%	6s	0.3084	6.46%	11s	0.3584	7.56%	32s
PMOCO	0.2775	1.84%	6s	0.3181	3.52%	10s	0.3678	5.13%	21s
CNH	0.2762	2.30%	7s	0.3182	3.49%	11s	0.3651	5.83%	22s
WE-CA	0.2796	1.09%	8s	0.3238	1.79%	14s	0.3756	3.12%	24s
PCDAM	**0.2827**	0.00%	10s	**0.3297**	0.00%	37s	**0.3877**	0.00%	160s



(a) E20 (b) E50 (c) E100

Fig. 2: Pareto Fronts on Synthetic Dataset.

TABLE II: Comparison Results on the gbd Benchmark Dataset.

Name	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	NSGA-II	EMNH	PMOCO	CNH	WE-CA	PCDAM
gbd-1	12	22	0.4407	0.4424	0.4581	0.4543	0.4557	**0.4600**
gbd-2	12	26	0.4457	0.4515	0.4528	**0.4574**	0.4559	0.4489
gbd-3	12	22	0.4757	0.4804	0.4750	0.4919	**0.4979**	0.4907
gbd-4	11	19	0.4420	0.4514	0.4405	0.4652	**0.4686**	0.4575
gbd-5	13	26	0.3581	0.3704	0.3829	0.3917	**0.3974**	0.3910
gbd-6	12	22	0.4348	0.4568	0.4444	**0.4617**	0.4306	0.4441
gbd-7	12	22	0.4248	0.4571	0.4578	0.4588	0.4659	**0.4684**
gbd-8	27	46	0.4514	0.4835	0.4981	**0.4987**	0.4866	0.4983
gbd-9	27	51	0.4978	0.5201	0.5296	0.5164	**0.5354**	0.5326
gbd-10	12	25	0.4397	0.5456	0.5427	0.5458	0.5510	**0.5511**
gbd-11	22	45	0.3022	0.4110	0.4036	**0.4260**	0.4180	0.4153
gbd-12	13	23	0.4397	0.3922	0.4280	0.4076	**0.4296**	0.4229
gbd-13	10	28	**0.3918**	0.3720	0.3653	0.3902	0.3828	0.3322
gbd-14	7	21	0.4359	0.4037	0.4501	0.4388	0.4298	**0.4627**
gbd-15	7	21	0.3960	0.4240	0.4288	0.4291	0.4149	**0.4520**
gbd-16	8	28	0.4060	0.4168	0.4198	0.4293	0.4253	**0.4509**
gbd-17	8	28	0.4789	0.4938	0.4040	0.5054	0.4840	**0.5218**
gbd-18	9	36	0.3953	0.4701	0.4905	0.4711	0.4711	**0.5101**
gbd-19	8	11	0.3673	0.3441	0.3697	0.3652	0.2793	**0.3783**
gbd-20	11	22	0.3982	0.4298	0.4256	0.4319	0.4305	**0.4335**
gbd-21	11	33	0.4378	0.4556	0.4651	0.4647	0.4761	**0.5008**
gbd-22	11	44	0.4351	0.4722	0.4495	0.4772	0.4851	**0.4963**
gbd-23	11	55	0.3715	0.3471	0.2807	0.3606	0.3463	**0.4013**
Mean	-	-	0.3981	0.4388	0.4005	0.4255	0.4112	**0.4528**
Gap	-	-	12.08%	3.09%	11.55%	6.03%	9.19%	0.00%

TABLE III: Comparison of the Generalization Ability Across Different Graph Structures.

Method	Small-world			Clustered		
	E20	E50	E100	E20	E50	E100
NSGA-II	0.2736 (3.42%)	0.2218 (21.71%)	0.1410 (54.43%)	0.2587 (2.41%)	0.2234 (31.51%)	0.1621 (57.59%)
EMNH	0.2696 (4.84%)	0.3183 (5.94%)	0.2780 (10.15%)	0.2524 (4.79%)	0.3035 (6.96%)	0.3503 (8.35%)
PMOCO	0.2773 (2.12%)	0.3275 (3.22%)	0.2765(10.63%)	0.2578 (2.75%)	0.3161(3.10%)	0.3593 (5.99%)
CNH	0.2729 (3.67%)	0.3271(3.34%)	0.2845 (8.05%)	0.2543 (4.07%)	0.3145 (3.59%)	0.3590 (6.07%)
WE-CA	0.2790 (1.52%)	0.3336 (1.42%)	0.2791 (9.79%)	0.2604 (1.77%)	0.3217 (1.38%)	0.3605 (5.68%)
PCDAM	**0.2833** (0.00%)	**0.3384** (0.00%)	**0.3094** (0.00%)	**0.2651** (0.00%)	**0.3262** (0.00%)	**0.3822** (0.00%)

showing clear advantages on larger graphs. This confirms that our model is robust and effectively transferable to diverse standard instances.

TABLE IV: Comparison Results in Large-Scale Instances.

Method	E150			E200		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
NSGA-II	0.0825	79.84%	31.6h	0.0382	90.34%	78.85h
EMNH	0.3573	12.71%	65S	0.3011	24.16%	113S
PMOCO	0.3720	9.11%	63s	0.3375	14.99%	106s
CNH	0.3679	10.12%	64s	0.3259	17.91%	111s
WE-CA	0.3476	15.08%	67s	0.1182	70.23%	121s
PCDAM	**0.4093**	0.00%	556s	**0.3970**	0.00%	1326s

C. Generalization

To comprehensively evaluate the robustness of PCDAM, we assess its generalization capability from two perspectives: distribution generalization across different graph topologies and size generalization on larger-scale instances.

1) Distribution Generalization: We evaluate structural generalization by directly applying models trained on uniform random graphs to small-world and clustered graphs, as reported in Table III. PCDAM demonstrates strong zero-shot transferability across different topologies. While baselines suffer severe performance degradation as the graph structure becomes more complex, exemplified by NSGA-II's gap deteriorating to over 57% and even the competitive WE-CA's gap widening to 9.79% on Small-world E100, PCDAM consistently achieves the best performance across all structural variants among the compared methods.

This resilience validates the effectiveness of our dynamic objective-aware features. By explicitly grounding decisions in real-time state information (e.g., accumulated cost D_2^t and current makespan D_3^t), the model learns a generalized reasoning logic that evaluates how each action affects the two conflicting objectives, rather than overfitting to static spatial patterns found in the training set. This design effectively decouples the policy from specific distribution characteristics, ensuring robustness across heterogeneous topologies.

2) Size Generalization: To evaluate zero-shot size generalization, we test models trained on small-scale instances ($|\mathcal{E}| = 100$) directly on larger datasets (E150 and E200), except CNH, which utilizes mixed-size training ($|\mathcal{E}| \in \{20, 30, \ldots, 100\}$) for explicit size adaptation. As shown in Table IV, traditional NSGA-II suffers from prohibitive computational costs and poor convergence, while WE-CA exhibits severe performance degradation on E200 due to overfitting to specific graph scales.

In contrast, PCDAM demonstrates superior robustness than the compared baselines: despite being restricted to single-size training, it significantly outperforms the size-aware CNH and other baselines, achieving the highest HV and a 0.00% gap on both E150 and E200. This advantage stems from our preference-aware attention mechanism, which constructs the query vector $\mathbf{Q}_c^t$ by incorporating the preference vector $\boldsymbol{\lambda}$ along with local context information. This design enables the attention mechanism to focus on relative relationships between candidate edges rather than absolute positional patterns, thereby facilitating effective transfer to larger problem scales.

D. Ablation Studies

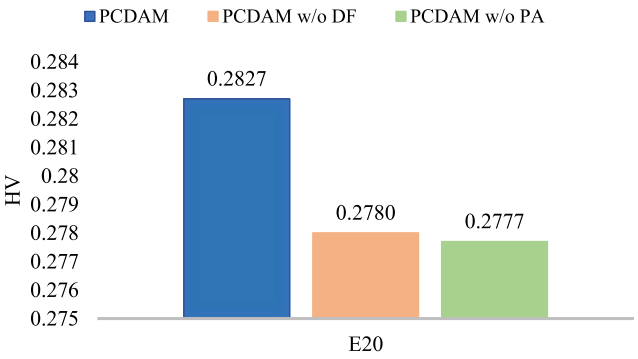


Fig. 3: Ablation Study on Key Components of PCDAM.

We conduct ablation studies on the E20 dataset to verify the effectiveness of two key components: Preference-Aware Attention (PA) and Dynamic Objective-Aware Features (DF). Specifically, we construct two variants: 1) **w/o PA**: removing $\boldsymbol{\lambda}$ from the context vector $\mathbf{h}_c^t$; 2) **w/o DF**: eliminating the dynamic feature branch that computes the correction bias $\mathbf{S}_2^t$. We compare them with the original PCDAM on E20 instances under identical settings

As shown in Fig. 3, removing either component leads to consistent performance degradation. Removing PA drops the HV to 0.2777, confirming that preference injection into query vectors is important for adaptive decision prioritization. Similarly, removing DF results in an HV of 0.2780, compared to the full model's 0.2827, demonstrating that real-time state

features (D_1^t, D_2^t, D_3^t) play an important role in evaluating objective trade-offs during decoding. These results indicate that both components play important and complementary roles in effective Pareto front approximation.

VI. Conclusions

In this paper, we proposed PCDAM, a novel single-model NCO framework for MOCARP. By leveraging two complementary components preference-aware attention and dynamic objective-aware features, our method successfully captured both preference-guided topological focus and real-time objective trade-offs during decoding. The integration of these mechanisms results in enhanced decision-making that adapts to user preferences while evaluating how each action affects conflicting objectives. Experimental results demonstrated that the proposed PCDAM outperformed representative evolutionary algorithms and existing NCO methods, achieving superior HV on a variety of MOCARP instances. Future work will explore stochastic dynamic routing scenarios, automated feature learning, and hybrid methods that combine learned policies with classical multi-objective optimization techniques.

References

[1] 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.

[2] B. L. Golden and R. T. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, no. 3, pp. 305–315, 1981.

[3] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "Evolutionary algorithms for periodic arc routing problems," *European Journal of Operational Research*, vol. 165, no. 2, pp. 535–553, 2005.

[4] M. Polacek, K. F. Doerner, R. F. Hartl, and V. Maniezzo, "A variable neighborhood search for the capacitated arc routing problem with intermediate facilities," *Journal of Heuristics*, vol. 14, no. 5, pp. 405–423, 2008.

[5] Y.-H. Jia, Q. Zheng, Y. Wang, Y. Mei, W.-N. Chen, and Z. Lin, "A neural solver with traversal-based feature representation and adjacent attention for capacitated arc routing problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 12, pp. 22 329–22 343, 2025.

[6] 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.

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

[8] 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.

[9] 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.

[10] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK report*, vol. 103, 2001.

[11] E. L. Ulungu and J. Teghem, "Multi-objective combinatorial optimization problems: A survey," *Journal of Multi-Criteria Decision Analysis*, vol. 3, no. 2, pp. 83–104, 1994.

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

[13] 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.

[14] Y. Zhang, J. Wang, Z. Zhang, and Y. Zhou, "Modrl/d-el: Multiobjective deep reinforcement learning with evolutionary learning for multiobjective optimization," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[15] J. Chen, J. Wang, Z. Zhang, Z. Cao, T. Ye, and S. Chen, "Efficient meta neural heuristic for multi-objective combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 36, pp. 56 825–56 837, 2023.

[16] 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*, vol. 34, no. 10, pp. 7978–7991, 2022.

[17] X. Lin, Z. Yang, and Q. Zhang, "Pareto set learning for neural multi-objective combinatorial optimization," *arXiv preprint arXiv:2203.15386*, 2022.

[18] J. Chen, Z. Cao, J. Wang, Y. Wu, H. Qin, Z. Zhang, and Y.-J. Gong, "Rethinking neural multi-objective combinatorial optimization via neat weight embedding," in *The Thirteenth International Conference on Learning Representations*, 2025.

[19] M. Ehrgott, X. Gandibleux, and A. Przybylski, "Exact methods for multi-objective combinatorial optimisation," in *Multiple Criteria Decision Analysis: State of the Art Surveys*. New York, NY: Springer New York, 2016, pp. 817–850.

[20] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "Moea/d with adaptive weight adjustment," *Evolutionary computation*, vol. 22, no. 2, pp. 231–264, 2014.

[21] Y. Yan, A. H. Chow, C. P. Ho, Y.-H. Kuo, Q. Wu, and C. Ying, "Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities," *Transportation Research Part E: Logistics and Transportation Review*, vol. 162, p. 102712, 2022.

[22] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

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

[24] J. Chen, Z. Zhang, Z. Cao, Y. Wu, Y. Ma, T. Ye, and J. Wang, "Neural multi-objective combinatorial optimization with diversity enhancement," *Advances in Neural Information Processing Systems*, vol. 36, pp. 39 176–39 188, 2023.

[25] J. Chen, J. Wang, Z. Cao, and Y. Wu, "Neural multi-objective combinatorial optimization via graph-image multimodal fusion," in *The Thirteenth International Conference on Learning Representations*, 2025.

[26] M. Fan, Y. Wu, Z. Cao, W. Song, G. Sartoretti, H. Liu, and G. Wu, "Conditional neural heuristic for multiobjective vehicle routing problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 3, pp. 4677–4689, 2024.

[27] Y. Wu, M. Fan, Z. Cao, R. Gao, Y. Hou, and G. Sartoretti, "Collaborative deep reinforcement learning for solving multi-objective vehicle routing problems," in *23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024*. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2024, pp. 1956–1965.

[28] R. Guo, F. Xue, A. Ming, and N. Sebe, "An efficient learning-based solver comparable to metaheuristics for the capacitated arc routing problem," *arXiv preprint arXiv:2403.07028*, 2024.

[29] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM journal on optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[30] J. S. DeArmon, "A comparison of heuristics for the capacitated Chinese postman problems," Master's thesis, University of Maryland, College Park, MD, USA, 1981.