GATING IS ATTENTION: A PLUG-AND-PLAY PREFERENCE-GATED DUAL-ATTENTION MODULE FOR NEURAL MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural multi-objective combinatorial optimization (MOCO) solves scalarized subproblems conditioned on preference vectors. Existing methods mainly use preferences for early representation fusion, weight-conditioned embedding, or route selection. However, they provide limited control over the attended features that directly guide decisions. We propose Preference-Gated Dual Attention (PGDA), a lightweight plug-and-play module that treats objective preferences as conditional controls over attention outputs. After standard attention, PGDA generates a gate from the context, the preference embedding, and the attention output. The gate then reweights the attended representation for preference-aware decision making. This modulation prioritizes preference-relevant features without altering instance representations. Experiments on multiple MOCO benchmarks show that PGDA provides consistent gains for single-model solvers and strong POCCO-based backbones. PGDA also achieves competitive solution quality with efficient inference, especially in cross-scale and large-scale generalization settings.

1 INTRODUCTION

Multi-objective combinatorial optimization (MOCO) is pervasive in real-world decision-making scenarios, such as production scheduling (Luo et al., 2021; Liu et al., 2023) and logistics planning (Zajac & Huber, 2021; Shao et al., 2023). Unlike single-objective combinatorial optimization, MOCO optimizes multiple conflicting objectives simultaneously and aims to identify Pareto-optimal solutions that characterize different trade-offs (Mavrotas et al., 2015; Tian et al., 2021). Traditional exact methods can in principle obtain complete Pareto-optimal sets, but their computational cost grows rapidly with problem size (Cao et al., 2013; Greco et al., 2016). Heuristic methods produce approximate Pareto fronts more efficiently, yet they often require extensive iterations, manual tuning, and problem-specific knowledge. Moreover, they usually restart the search from scratch for each instance, limiting their efficiency and transferability (Bengio et al., 2021; Kallestad et al., 2023).

Recently, neural combinatorial optimization has shown strong potential by learning reusable decision patterns from data. It has achieved remarkable progress in single-objective combinatorial optimization (Kool et al., 2019; Kwon et al., 2020; Hottung et al., 2022; Li et al., 2022; Wu et al., 2022; Grinsztajn et al., 2023; Luo et al., 2024; Son et al., 2025), and has further stimulated neural MOCO methods (Li et al., 2021; Zhang et al., 2023; Lin et al., 2022; Chen et al., 2023b;a; Shao et al., 2023; Wang et al., 2024; Chen et al., 2025a;b; Fan et al., 2025; 2026). A common paradigm is to decompose a MOCO problem into scalarized subproblems, each associated with a preference vector, and then solve these subproblems using neural policies. Early methods train or fine-tune separate models for different subproblems, leading to high training cost and limited generalization (Li et al., 2021; Zhang et al., 2023; Chen et al., 2023a). Recent single-model methods instead learn a unified policy conditioned on preferences, for example through hypernetworks (Lin et al., 2022), cross-attention between preference and instance embeddings (Fan et al., 2025; Wang et al., 2024), or direct weight embedding into node representations (Chen et al., 2025a). More recently, POCCO (Fan et al., 2026), a state-of-the-art neural MOCO method, further improves performance by introducing conditional computation and pairwise solution-preference learning.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Despite recent progress, preference modeling in neural MOCO remains limited. Existing methods mainly incorporate preferences through early representation fusion, additional tokens, preference-conditioned embeddings, or route selection. These designs offer limited control over attended features that guide decisions, while early preference signals may weaken during deep encoding or interfere with instance representations. We therefore propose directly modulating attended features with preferences to reflect the target trade-off while preserving instance structure.

Motivated by the effectiveness of gating as an attention-like modulation mechanism (Qiu et al., 2025), we propose Preference-Gated Dual Attention (PGDA), a lightweight plug-and-play module for neural MOCO. PGDA applies preference-guided gating after standard attention, forming a two-stage mechanism in which attention captures instance-dependent interactions and gating refines aggregated features for preference-aware decision making. PGDA can be integrated into neural MOCO backbones, including CNH (Fan et al., 2025), WE-CA (Chen et al., 2025a), and POCCO-based solvers (Fan et al., 2026). Extensive experiments demonstrate consistent improvements across strong backbones, particularly under zero-shot cross-scale and large-scale generalization.

Our contributions are summarized as follows: (1)We propose PGDA, a lightweight module that can be integrated into various neural MOCO backbones to enhance preference-conditioned decision making. (2)We introduce a preference-gated dual-attention mechanism, where the gate is jointly determined by the context, preference representation, and attention output. This provides a simple post-attention extension to standard QKV attention. (3)Experiments on multiple MOCO benchmarks show that PGDA improves single-model solvers and POCCO-based state-of-the-art solvers in most settings, while maintaining efficient inference and strong generalization under scale shifts.

2 Related works

2.1 Traditional MOCO methods

Traditional MOCO methods can be broadly divided into exact algorithms and heuristic approaches. Exact algorithms, such as branch-and-bound and dynamic programming variants, aim to recover complete Pareto-optimal sets (Fazlollahi et al., 2012; Cao et al., 2013), but their computational cost becomes prohibitive as the problem size increases (Greco et al., 2016). Heuristic methods, including MOEAs such as NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2001), MOEA/D (Zhang & Li, 2007), MOACO (García-Martínez et al., 2007), and MOPSO (Borhanazad et al., 2014), as well as local search methods (Jaszkiewicz, 2002; Paquete & Stützle, 2003; Shaw, 1998; Hansen & Mladenović, 2001; Ropke & Pisinger, 2006), can efficiently approximate Pareto fronts. However, they often require many iterations, problem-specific operators, and careful parameter tuning.

2.2 Neural MOCO methods

Neural MOCO methods usually follow the decomposition paradigm and solve scalarized subproblems conditioned on preference vectors. Multi-model methods assign a dedicated solver to each subproblem and are often trained via transfer learning (Li et al., 2021) or meta-learning (Zhang et al., 2023; Chen et al., 2023a). In contrast, single-model methods train a unified policy for different preferences. PMOCO uses hypernetworks to generate weight-dependent parameters (Lin et al., 2022); CNH and MORAM model preference-instance interactions through cross-attention or related fusion mechanisms (Fan et al., 2025; Wang et al., 2024); WE-CA directly embeds objective weights into node representations (Chen et al., 2025a). More recently, POCCO (Fan et al., 2026) improves this line of work by introducing conditional computation and pairwise solution-preference learning. Specifically, it routes different subproblems through specialized computation paths and trains the policy using winner–loser solution comparisons. While effective, these methods mainly use preferences for parameter generation, feature fusion, route selection, or training supervision, rather than explicitly modulating the attention outputs used for decision making.

3 Preliminaries

3.1 MOCO Definition

Without loss of generality, we consider a minimization-type multi-objective combinatorial optimization (MOCO) problem, where a decision vector x is selected from a discrete feasible space Ω and

mapped to an m-dimensional objective vector $F(x) \in \mathbb{R}^m$. The problem is formulated as

$$\min_{x \in \Omega} F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^\top, \tag{1}$$

where $m \geq 2$ and each objective function $f_i : \Omega \to \mathbb{R}$ is to be minimized.

For two feasible solutions $x_a, x_b \in \Omega$, x_a Pareto dominates x_b, denoted by $x_a \prec x_b$, if $f_i(x_a) \leq f_i(x_b)$ for all $i = 1, \ldots, m$ and $f_j(x_a) < f_j(x_b)$ for at least one objective j. A feasible solution $x^* \in \Omega$ is Pareto optimal if it is not dominated by any other feasible solution. The set of all Pareto-optimal solutions is called the Pareto set, and its image in the objective space is the Pareto front:

$$PS = \{x^* \in \Omega \mid \nexists x \in \Omega,\ x \prec x^*\}, PF = \{F(x) \mid x \in PS\}. \tag{2}$$

3.2 Decomposition-based Neural MOCO

Decomposition is a widely used strategy for solving MOCO problems (Zhang & Li, 2007). It converts the original MOCO into a set of scalarized subproblems, each associated with a preference vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$, where $\lambda_i \geq 0$ and $\sum_{i=1}^m \lambda_i = 1$. Given a feasible solution or solution sequence π, the weighted-sum scalarization is defined as

$$g_{\text{ws}}(\pi \mid \boldsymbol{\lambda}) = \sum_{i=1}^{m} \lambda_i f_i(\pi). \tag{3}$$

Other scalarization functions can also be adopted, but the essential idea is to obtain preference-conditioned scalar subproblems.

In neural MOCO, solution construction is usually modeled as a sequential decision-making process. A neural policy p_θ generates a solution sequence $\pi = (\pi_1, \ldots, \pi_T)$ conditioned on an instance s and a preference vector λ:

$$p_\theta(\pi \mid \boldsymbol{\lambda}, s) = \prod_{t=1}^{T} p_\theta(\pi_t \mid \boldsymbol{\lambda}, \pi_{1:t-1}, s). \tag{4}$$

The preference vector therefore guides the policy toward different trade-offs among objectives, making preference representation and utilization central to neural MOCO.

4 Methodology

4.1 Overall Framework

Figure 1 illustrates the proposed architecture. Given a MOCO instance, the encoder maps raw node features to latent embeddings and refines them through L attention layers equipped with PGDA. By conditioning the representations on the preference vector, the encoder captures both instance structure and the target objective trade-off. PGDA is integrated into the decoder to support sequential solution construction. At each decoding step, the decoder attends to the encoded node representations and generates a context-aware representation. PGDA then modulates this representation based on the decoding context and target preference. This preference-guided modulation directs node selection toward solutions that are consistent with the specified trade-off. The proposed architecture therefore combines structure-aware representation learning with preference-aware decision-making, allowing the model to flexibly generate solutions for different preference vectors.

4.2 Preference-Gated Dual Attention

PGDA introduces preference information as a conditional control signal rather than directly injecting it into node representations. Given an M-dimensional preference vector $\boldsymbol{\lambda}$, we first project it into the hidden feature space:

$$\mathbf{h}_\lambda = \text{FFN}_\lambda(\boldsymbol{\lambda}) \tag{5}$$

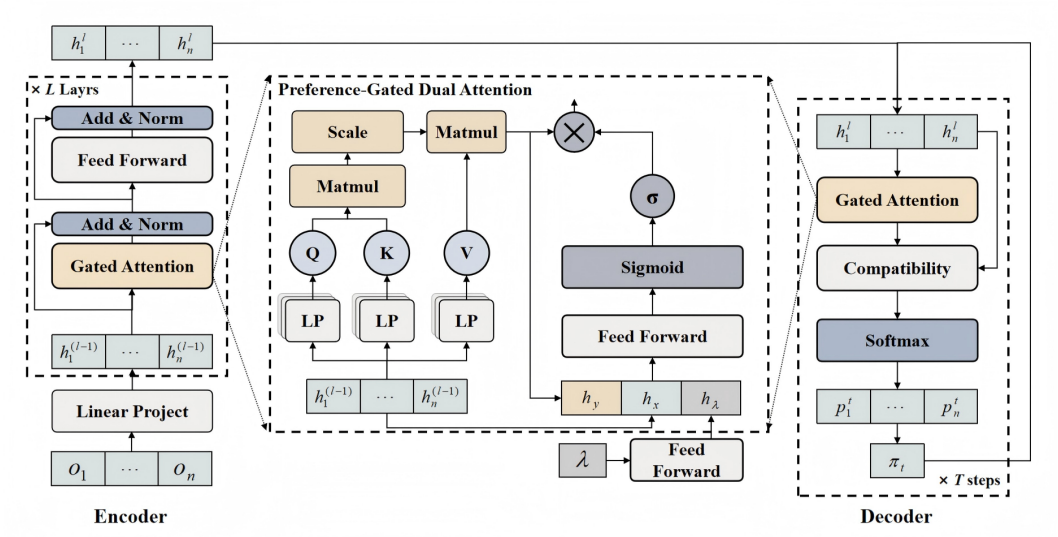

Figure 1: Overall architecture of the PGDA-based neural MOCO model. The encoder refines node representations through stacked PGDA-enhanced layers, while the decoder constructs solutions conditioned on a target preference vector. The middle panel illustrates PGDA, where a dynamic gate derived from X, Pref, and Y modulates the standard attention output.

At the l-th encoder layer, standard self-attention is first used to aggregate structural information among nodes:

$$\mathbf{y}_i^{(l)} = \mathrm{Attn}\left(\mathbf{h}_i^{(l-1)}\mathbf{H}^{(l-1)}\right), \tag{6}$$

where $\mathbf{H}^{(l-1)} = \{\mathbf{h}_1^{(l-1)}, \ldots, \mathbf{h}_n^{(l-1)}\}$ denotes the input node representations of the previous layer. Based on the attention output, PGDA generates a preference-aware gate and uses it to modulate the aggregated feature:

$$\mathbf{g}_i^{(l)} = \sigma\left(\mathrm{FFN}_{en}\left([\mathbf{h}_i^{(l-1)}; \mathbf{h}_\lambda; \mathbf{y}_i^{(l)}]\right)\right), \quad \hat{\mathbf{h}}_i^{(l)} = \mathbf{y}_i^{(l)} \odot \mathbf{g}_i^{(l)} \tag{7}$$

In this way, PGDA decouples instance-level feature aggregation from preference-guided feature filtering. The attention module first captures dependencies among nodes, while the dynamic gate selectively moderate the aggregated features according to the desired trade-off.

The mechanism is also applied in the decoder. Let $\mathbf{x}_t$ denote the decoding context at step t, and $\mathbf{H}$ be the encoded candidate representations. The decoder first produces a context-aware attention output: $\mathbf{y}_t = \mathrm{Attn}(\mathbf{x}_t, \mathbf{H})$. Then, PGDA modulates the output through a preference-aware gate:

$$\hat{\mathbf{y}}_t = \mathbf{y}_t \odot \sigma\left(\mathrm{FFN}_{de}\left([\mathbf{x}_t; \mathbf{h}_\lambda; \mathbf{y}_t]\right)\right) \tag{8}$$

The modulated representation $\hat{\mathbf{y}}_t$ is used to score candidate nodes and compute selection probabilities. PGDA thus enables lightweight, fine-grained preference adaptation in both encoder and decoder representations.

4.3 Training and Inference

Given a preference vector $\boldsymbol{\lambda}$ and an instance s, the model defines a conditional policy $p_\theta(\pi \mid \boldsymbol{\lambda}, s)$ to generate a solution π. A scalarization function maps the multi-objective cost vector to a scalar value. For fair comparison, we use weighted-sum (WS) scalarization for all neural methods. Let $C(\pi \mid \boldsymbol{\lambda}, s)$ denote the scalarized cost of solution π under preference vector $\boldsymbol{\lambda}$. Depending on the backbone, PGDA is trained using either REINFORCE or pairwise preference-based learning.

REINFORCE-based training. For the basic PGDA model and non-POCCO backbones, we train a unified policy network using REINFORCE with a shared baseline to cover the continuous preference space. The reward is defined as

$$R(\pi \mid \boldsymbol{\lambda}, s) = -C(\pi \mid \boldsymbol{\lambda}, s). \tag{9}$$

The training objective is to maximize the expected reward over the preference distribution P_λ, yielding the following policy loss:

$$\mathcal{L}_{\mathrm{RL}}(\theta) = \mathbb{E}_{\boldsymbol{\lambda}\sim P_\lambda}\mathbb{E}_{\pi\sim p_\theta(\cdot|\boldsymbol{\lambda},s)}\Big[\big(R(\pi \mid \boldsymbol{\lambda}, s) - B(\boldsymbol{\lambda}, s)\big) \cdot \log p_\theta(\pi \mid \boldsymbol{\lambda}, s)\Big]. \tag{10}$$

To reduce the variance of gradient estimation and improve training stability, we use a weight-dependent shared baseline $B(\boldsymbol{\lambda}, s)$. In practice, $B(\boldsymbol{\lambda}, s)$ is computed as the mean reward of trajectories generated from different starting nodes within the same mini-batch.

Pairwise preference-based training. For POCCO-based variants, including POCCO-C, POCCO-W, POCCO-C-PGDA, and POCCO-W-PGDA, we follow the pairwise solution-preference learning strategy of POCCO (Fan et al., 2026). For each scalarized subproblem $(\boldsymbol{\lambda}, s)$, the policy samples two candidate solutions, π^a and π^b, and ranks them by scalarized cost. The better and worse solutions are denoted as π^w and π^l, respectively:

$$\pi^w = \arg\min_{\pi\in\{\pi^a,\pi^b\}} C(\pi \mid \boldsymbol{\lambda}, s), \pi^l = \arg\max_{\pi\in\{\pi^a,\pi^b\}} C(\pi \mid \boldsymbol{\lambda}, s). \tag{11}$$

Instead of raw scalarized costs, each sampled solution is assigned a normalized log-likelihood score:

$$r_\theta(\pi \mid \boldsymbol{\lambda}, s) = \frac{1}{|\pi|}\sum_{t=1}^{|\pi|} \log p_\theta(\pi_t \mid \pi_{<t}, \boldsymbol{\lambda}, s). \tag{12}$$

The pairwise preference loss is

$$\mathcal{L}_{\mathrm{pair}}(\theta) = -\mathbb{E}_{\boldsymbol{\lambda}\sim P_\lambda}\mathbb{E}_{\pi^{a,b},\sim p_\theta(\cdot|\boldsymbol{\lambda},s)}\Big[\log\sigma\Big(\beta\big(r_\theta(\pi^w \mid \boldsymbol{\lambda}, s) - r_\theta(\pi^l \mid \boldsymbol{\lambda}, s)\big)\Big)\Big]. \tag{13}$$

where the inner expectation is over two independently sampled solutions, $\pi^a, \pi^b \sim p_\theta(\cdot \mid \boldsymbol{\lambda}, s)$, which are ranked as π^w and π^l by the scalarized costs in Eq.(11). The parameter $\beta > 0$ controls the sharpness of preference discrimination. This objective encourages the policy to assign a higher likelihood score to the preferred solution.

Inference. During inference, the model takes preference vectors and solves the corresponding preference-conditioned subproblems in parallel. Each preference vector produces a Pareto candidate through a single forward pass, and the resulting candidates collectively form a high-quality approximate Pareto front. With PGDA, the unified policy generalizes robustly to unseen scales.

5 Experiment

5.1 Experimental Settings

Benchmarks. To evaluate the generality and effectiveness of PGDA, we consider three representative MOCO benchmarks: the multi-objective traveling salesman problem (MOTSP) (Lust & Teghem, 2010a), the multi-objective capacitated vehicle routing problem (MOCVRP) (Zajac & Huber, 2021), and the multi-objective knapsack problem (MOKP) (Ishibuchi et al., 2015). Following the standard setting in neural combinatorial optimization (Kwon et al., 2020), node coordinates are uniformly sampled from the unit square $[0, 1]^2$ for routing problems. More details are provided in the appendix. During training, instances are randomly generated with problem sizes $n \in \{20, \ldots, 100\}$, except for Bi-KP, where $n \in \{50, \ldots, 200\}$. During testing, each instance is assigned a set of uniformly distributed preference vectors to construct an approximate Pareto front. Unless otherwise specified, we use 101 preference vectors for bi-objective problems and 105/1035/10011 preference vectors for tri-objective problems. Experiments used Ubuntu 22.04, six 32-GB GPUs, and 72 Intel(R) Xeon(R) Platinum 8375C vCPUs, with Python 3.12, PyTorch 2.7.0, and CUDA 12.8.

Compared methods. We compare PGDA with three groups of baselines. All neural methods adopt weighted-sum (WS) scalarization to ensure a fair comparison. (1) *Single-model neural MOCO methods*, including CNH (Fan et al., 2025), WE-CA (Chen et al., 2025a), POCCO-C, and POCCO-W (Fan et al., 2026). To verify the plug-and-play property of PGDA, we evaluate PGDA-enhanced variants of all four backbones. The results for POCCO-C-PGDA and POCCO-W-PGDA are reported in the main text, while those for CNH-PGDA and WE-CA-PGDA are provided in Appendix G. (2) *Multi-model neural MOCO methods*, including MDRL (Zhang et al., 2023) and EMNH (Chen et al., 2023a), which fine-tune subproblem-specific models from a shared meta-model. (3) *Non-learning methods*, including MOEA/D (Zhang & Li, 2007), NSGA-II (Deb et al., 2002), MOGLS (Jaszkiewicz, 2002), and PPLS/D-C (Shi et al., 2024). For fairness, MOEA/D and NSGA-II are run for 4000 iterations, MOGLS is run for 4000 iterations with 100 local-search steps per iteration, and PPLS/D-C is run for 200 iterations. For specialized WS-based solvers, we additionally include WS-LKH for MOTSP (Helsgaun, 2000; Tinós et al., 2018) and WS-DP for MOKP.

Training details. For a fair comparison, PGDA-enhanced models follow the training protocol of their corresponding backbones. The basic PGDA model and PGDA-enhanced non-POCCO backbones are trained using REINFORCE with a shared baseline. POCCO-W/C and their PGDA-enhanced variants are trained using pairwise preference-based learning. Unless otherwise stated, all neural models are trained for 200 epochs, with 100,000 dynamically generated instances per epoch. The batch size is set to 64. We use the Adam optimizer with a learning rate of 10^{-4} for REINFORCE-based methods and 3×10^{-4} for POCCO-based methods, and a weight decay of 10^{-6}. Preference vectors are generated using the Das–Dennis scheme.

Evaluation protocol. We adopt hypervolume (HV) as the evaluation metric (Audet et al., 2021), since it captures both convergence and diversity of the solution set. Higher HV indicates a better approximate Pareto front. Except for the single-instance KroAB benchmarks, all solution-quality results are averaged over 200 distinct test instances, reducing sensitivity to instance randomness and providing a robust evaluation. Methods with the suffix "-Aug" use instance augmentation (Lin et al., 2022). In addition to solution quality, we report total inference time over the same test instances. Statistical significance is assessed using the two-sided Wilcoxon signed-rank test at the 1% level. In tables, the best and second-best results are marked in **bold** and underline, respectively.

5.2 Experimental results

Comparison analysis. Tables 1 and 2, together with Figure 2, summarize PGDA performance across diverse MOCO benchmarks. Overall, PGDA achieves competitive solution quality with efficient inference. On Bi-TSP, its benefits become more evident as problem size increases. Although POCCO-W-PGDA performs slightly worse than POCCO-W on Bi-TSP20 and Bi-TSP50, this may be because limited structural information in small instances causes preference-guided gating to overemphasize preference-aligned features and reduce solution-set diversity. On Bi-CVRP, PGDA consistently improves single-model and POCCO-based backbones, with POCCO-W-PGDA-Aug achieving the best or tied-best HV across all tested sizes. On Bi-KP, PGDA remains competitive despite small differences among strong methods, and POCCO-C-PGDA matches the best HV on Bi-KP100 and Bi-KP200, indicating that PGDA also generalizes to combinatorial selection problems. Figure 2 further shows that PGDA-enhanced variants maintain competitive HV on tri-objective TSP under different numbers of preference vectors. Overall, these results support the effectiveness and general applicability of PGDA across problem structures, scales, backbones, and objective counts.

Out-of-distribution size generalization. We further evaluate representative methods on large-scale Bi-TSP150/200 and real-world KroAB100/150/200 instances (Lust & Teghem, 2010b). Complete results are provided in Appendix F. As shown in Figure 3, POCCO-W-PGDA consistently achieves the highest HV among neural methods on KroAB100/150/200. Integrating PGDA into POCCO-W increases HV from 0.7006 to 0.7012, 0.6976 to 0.6982, and 0.7369 to 0.7379 on KroAB100/150/200, respectively. These results indicate that PGDA improves the robustness and generalization of neural MOCO policies under both scale and distribution shifts.

Ablation Study. The comparisons between POCCO-C/W and POCCO-C/W-PGDA demonstrate the benefits of integrating PGDA. Figure 4 further examines its key components on Bi-TSP. Remov-

Table 1: Comparison results on routing-based bi-objective MOCO benchmarks.

	Method	Bi-TSP20			Bi-TSP50			Bi-TSP100		
		HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	WS-LKH	0.6270	0.02%	10m	0.6415	-0.05%	1.8h	**0.7090**	**-0.33%**	6.0h
	MOGLS	**0.6279**	**-0.12%**	1.6h	0.6330	1.28%	3.7h	0.6854	3.01%	11h
	PPLS/D-C	0.6256	0.24%	26m	0.6282	2.02%	2.8h	0.6844	3.15%	11h
	MOEA/D	0.6241	0.48%	1.7h	0.6316	1.49%	1.8h	0.6899	2.37%	2.2h
	NSGA-II	0.6258	0.21%	6.0h	0.6120	4.55%	6.1h	0.6692	5.30%	6.9h
Multi-model	MDRL	0.6271	0.00%	5s	0.6364	0.75%	9s	0.6969	1.38%	17s
	EMNH	0.6271	0.00%	5s	0.6364	0.75%	9s	0.6969	1.38%	**16s**
Single-model	CNH	0.6270	0.02%	14s	0.6387	0.39%	17s	0.7019	0.67%	29s
	WE-CA	0.6270	0.02%	7s	0.6392	0.31%	10s	0.7034	0.46%	21s
	PGDA	0.6271	0.00%	**4s**	0.6400	0.18%	**8s**	0.7045	0.31%	25s
POCCO-based	-C	0.6275	-0.06%	14s	0.6409	0.04%	20s	0.7047	0.28%	42s
	-W	0.6275	-0.06%	7s	0.6411	0.01%	14s	0.7055	0.16%	36s
	-C-PGDA	0.6274	-0.04%	13s	0.6413	-0.02%	21s	0.7059	0.11%	56s
	-W-PGDA	0.6274	-0.04%	14s	0.6410	0.03%	23s	0.7060	0.09%	57s
Multi-model-Aug	MDRL-Aug	0.6271	0.00%	33s	0.6408	0.06%	1.7m	0.7022	0.63%	14m
	EMNH-Aug	0.6271	0.00%	33s	0.6408	0.06%	1.7m	0.7023	0.62%	14m
Single-model-Aug	CNH-Aug	0.6271	0.00%	1.3m	0.6410	0.03%	3.9m	0.7054	0.18%	12m
	WE-CA-Aug	0.6271	0.00%	1.3m	0.6413	-0.02%	3.6m	0.7066	0.01%	12m
	PGDA-Aug	0.6270	0.02%	1.3m	0.6414	-0.03%	3.6m	0.7070	-0.05%	12m
POCCO-based-Aug	-C-Aug	0.6270	0.02%	2.2m	0.6416	-0.07%	4.0m	0.7071	-0.06%	14m
	-W-Aug	0.6270	0.02%	2.2m	**0.6418**	**-0.10%**	4.0m	0.7078	-0.16%	14m
	-C-PGDA-Aug	0.6270	0.02%	2.2m	0.6416	-0.07%	4.0m	0.7077	-0.15%	14m
	-W-PGDA-Aug	0.6270	0.02%	2.2m	0.6417	-0.08%	4.0m	0.7080	-0.19%	14m
	Method	Bi-CVRP20			Bi-CVRP50			Bi-CVRP100		
		HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	MOEA/D	0.4255	1.07%	2.3h	0.4000	2.53%	2.9h	0.3953	3.16%	5.0h
	NSGA-II	0.4275	0.60%	6.4h	0.3896	5.06%	8.8h	0.3620	11.32%	9.4h
	MOGLS	0.4278	0.54%	9.0h	0.3984	2.92%	20h	0.3875	5.07%	72h
	PPLS/D-C	0.4287	0.33%	1.6h	0.4007	2.36%	9.7h	0.3946	3.33%	38h
Multi-model	MDRL	0.4291	0.23%	8s	0.4082	0.53%	13s	0.4056	0.64%	32s
	EMNH	0.4299	0.05%	7s	0.4098	0.14%	13s	0.4072	0.24%	31s
Single-model	CNH	0.4287	0.33%	11s	0.4087	0.41%	15s	0.4065	0.42%	25s
	WE-CA	0.4290	0.26%	7s	0.4089	0.36%	**10s**	0.4068	0.34%	**21s**
	PGDA	0.4290	0.26%	**6s**	0.4093	0.26%	**10s**	0.4075	0.17%	26s
POCCO-based	-C	0.4294	0.16%	16s	0.4101	0.07%	25s	0.4079	0.07%	53s
	-W	0.4294	0.16%	8s	0.4102	0.04%	17s	0.4084	-0.05%	46s
	-C-PGDA	0.4295	0.14%	10s	0.4103	0.02%	25s	0.4081	0.02%	54s
	-W-PGDA	0.4295	0.14%	10s	0.4103	0.02%	24s	0.4085	-0.07%	60s
Multi-model-Aug	MDRL-Aug	0.4294	0.16%	11s	0.4092	0.29%	36s	0.4072	0.24%	2.8m
	EMNH-Aug	**0.4302**	**-0.02%**	11s	0.4106	-0.05%	35s	0.4079	0.07%	2.8m
Single-model-Aug	CNH-Aug	0.4299	0.05%	21s	0.4101	0.07%	45s	0.4077	0.12%	1.9m
	WE-CA-Aug	0.4300	0.02%	15s	0.4103	0.02%	36s	0.4081	0.02%	1.8m
	PGDA-Aug	0.4300	0.02%	16s	0.4105	-0.03%	58s	0.4084	-0.05%	2m
POCCO-based-Aug	-C-Aug	**0.4302**	**-0.02%**	31s	0.4108	-0.10%	1.4m	0.4086	-0.10%	2.4m
	-W-Aug	0.4301	0.00%	24s	0.4108	-0.10%	1.2m	0.4089	-0.17%	2.3m
	-C-PGDA-Aug	**0.4302**	**-0.02%**	25s	**0.4109**	**-0.13%**	1.3m	0.4087	-0.12%	2.4m
	-W-PGDA-Aug	**0.4302**	**-0.02%**	23s	**0.4109**	**-0.13%**	1.3m	**0.4090**	**-0.20%**	2.4m

ing X or Y, or disabling gating in either the encoder or decoder, consistently degrades performance. These results indicate that PGDA benefits from jointly incorporating context, attended features, and preference-conditioned modulation in both encoding and decoding.

Optimality on Scalarized Subproblems. We further examine the ability of learned policies to optimize individual scalarized subproblems. Table 3 reports results on Bi-TSP100 under three rep-

Table 2: Comparison results on bi-objective knapsack-based MOCO benchmarks.

	Method	Bi-KP50			Bi-KP100			Bi-KP200		
		HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
Non-learnable	WS-DP	0.3561	0.00%	22m	0.4532	0.02%	2.0h	0.3601	0.00%	5.8h
	MOEA/D	0.3540	0.59%	1.6h	0.4508	0.55%	1.7h	0.3581	0.56%	1.8h
	NSGA-II	0.3547	0.39%	7.8h	0.4520	0.28%	8.0h	0.3590	0.31%	8.4h
	MOGLS	0.3540	0.59%	5.8h	0.4510	0.51%	10h	0.3582	0.53%	18h
	PPLS/D-C	0.3528	0.93%	18m	0.4480	1.17%	47m	0.3541	1.67%	1.5h
Multi-model	MDRL	0.3530	0.87%	**6s**	0.4532	0.02%	21s	0.3601	0.00%	1.2m
	EMNH	0.3561	0.00%	**6s**	**0.4535**	**-0.05%**	21s	**0.3603**	**-0.06%**	1.2m
Single-model	PMOCO	0.3552	0.25%	9s	0.4523	0.22%	19s	0.3595	0.17%	1.3m
	CNH	0.3556	0.14%	16s	0.4527	0.13%	23s	0.3598	0.08%	55s
	WE-CA	0.3558	0.08%	8s	0.4531	0.04%	**16s**	0.3602	-0.03%	**50s**
	PGDA	0.3557	0.11%	**6s**	0.4532	0.02%	18s	0.3601	0.00%	60s
POCCO-based	-C	0.3560	0.03%	20s	**0.4535**	**-0.05%**	36s	**0.3603**	**-0.06%**	1.4m
	-W	**0.3562**	**-0.03%**	11s	0.4534	-0.02%	26s	**0.3603**	**-0.06%**	1.3m
	-C-PGDA	0.3561	0.00%	16s	**0.4535**	**-0.05%**	29s	**0.3603**	**-0.06%**	1.2m
	-W-PGDA	0.3561	0.00%	10s	0.4531	0.04%	25s	**0.3603**	**-0.06%**	1.2m

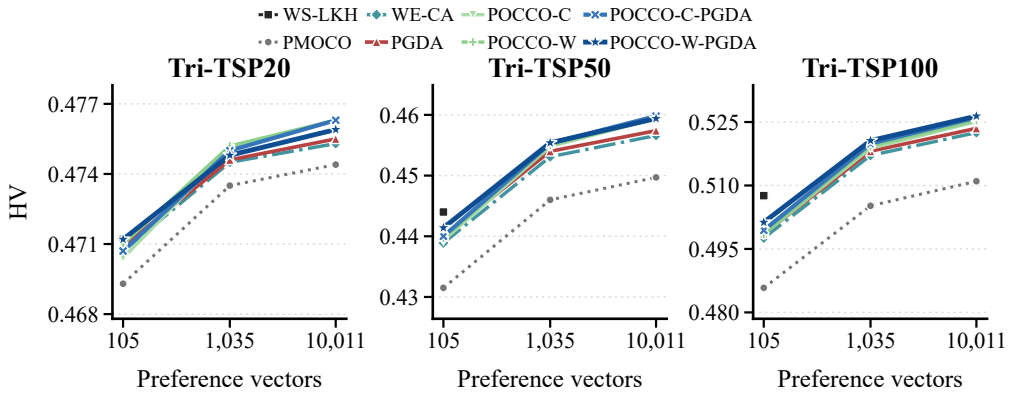

Figure 2: HV results on tri-objective TSP benchmarks under different numbers of preference vectors.

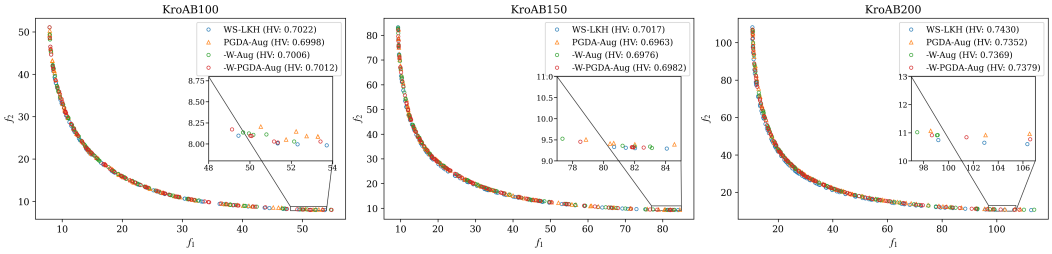

Figure 3: Pareto front comparison on KroAB100/150/200. The zoomed-in views further show improved solution coverage and boundary quality in local trade-off regions.

resentative preference vectors. PGDA-Aug obtains the best result under the balanced preference $(0.5, 0.5)$. Moreover, integrating PGDA into POCCO-W consistently improves all three preferences, reducing the Gap from 0.25% to 0.15%, from 0.79% to 0.28%, and from 0.24% to 0.14%, respectively. The improvement is evident under the balanced preference, which requires objective trade-offs. These results complement the Pareto-front evaluation by showing that PGDA improves both approximate Pareto-front quality and optimization accuracy for specific preferences.

Table 3: Scalarized performance on Bi-TSP100 under three test preference vectors.

Method	$g(\pi \mid (1,0)) \downarrow$		$g(\pi \mid (0.5,0.5)) \downarrow$		$g(\pi \mid (0,1)) \downarrow$	
	Value	Gap↓	Value	Gap↓	Value	Gap↓
LKH	**7.7632**	**0.00%**	17.3094	0.00%	**7.7413**	**0.00%**
POMO-Aug	7.7659	0.03%	17.4421	0.77%	7.7716	0.39%
MDRL-Aug	8.0316	3.46%	17.6209	1.80%	8.0290	3.72%
EMNH-Aug	8.0620	3.85%	17.5979	1.67%	8.0439	3.91%
PMOCO-Aug	8.1401	4.85%	17.5723	1.52%	8.1202	4.89%
WE-CA-Aug	7.8198	0.73%	17.4633	0.89%	7.7937	0.68%
PGDA-Aug	7.7994	0.47%	**17.2854**	**-0.14%**	7.7775	0.47%
POCCO-W-Aug	7.7827	0.25%	17.4460	0.79%	7.7602	0.24%
POCCO-W-PGDA-Aug	7.7751	0.15%	17.3574	0.28%	7.7523	0.14%

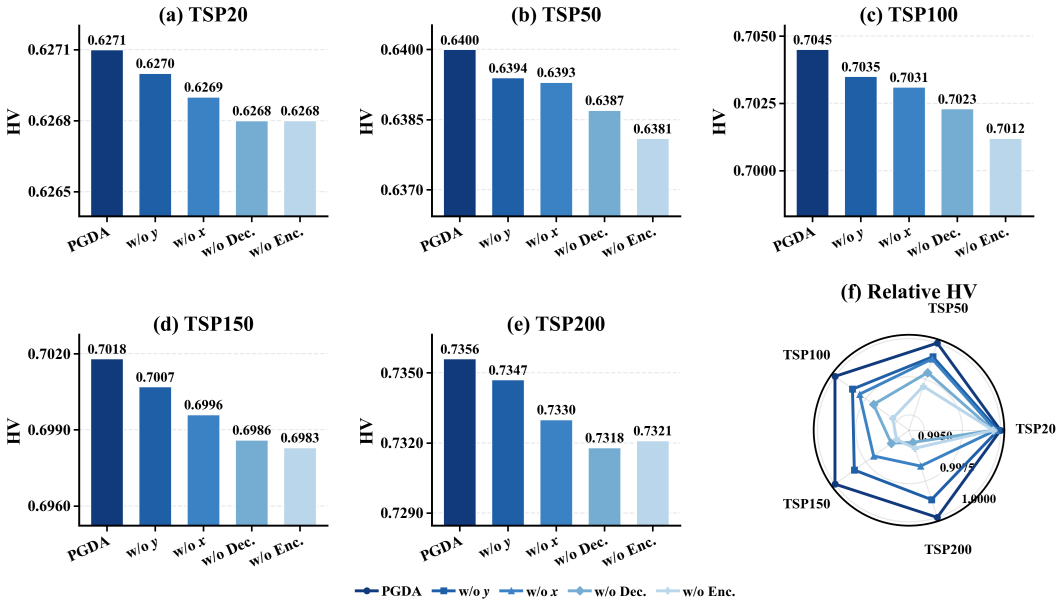

Figure 4: Ablation results of PGDA on Bi-TSP. "w/o x", "w/o y", "w/o en", and "w/o de" remove the context input, attention output, encoder gate, and decoder gate, respectively.

6 Conclusion

In this paper, we propose Preference-Gated Dual Attention (PGDA), a lightweight plug-and-play module for neural multi-objective combinatorial optimization. PGDA modulates attention outputs through a dynamic gate conditioned on the context, preference representation, and attended features, enabling preference-aware refinement while preserving instance representations. It can be integrated into single-model solvers and POCCO-based backbones with minimal changes. Experiments on routing, knapsack, large-scale, and real-world benchmarks show that PGDA improves strong neural solvers in most settings while maintaining efficient inference and generalization across problem structures, scales, and objective dimensions. Although some gains are modest, they are achieved on already strong backbones. More importantly, PGDA is a general module that can improve different neural solvers without substantial architectural changes. Ablations further confirm that the gains arise from preference-aware gated modulation rather than increased parameter count. Future work will explore more adaptive preference modeling and broader multi-objective applications.

REFERENCES

Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. Performance indicators in multiobjective optimization. *European Journal of Operational Research*, 292(2):397–422, July 2021. ISSN 0377-2217. doi: 10.1016/j.ejor.2020.11.016.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, April 2021. ISSN 0377-2217. doi: 10.1016/j.ejor.2020.07.063.

Hanieh Borhanazad, Saad Mekhilef, Velappa Gounder Ganapathy, Mostafa Modiri-Delshad, and Ali Mirtaheri. Optimization of micro-grid system using mopso. *Renewable Energy*, 71:295–306, November 2014. ISSN 0960-1481. doi: 10.1016/j.renene.2014.05.006.

Dingzhou Cao, Alper Murat, and Ratna Babu Chinnam. Efficient exact optimization of multi-objective redundancy allocation problems in series-parallel systems. *Reliability Engineering & System Safety*, 111:154–163, March 2013. ISSN 0951-8320. doi: 10.1016/j.ress.2012.09.013.

Jinbiao Chen, Jiahai Wang, Zizhen Zhang, Zhiguang Cao, Te Ye, and Siyuan Chen. Efficient meta neural heuristic for multi-objective combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 56825–56837, October 2023a.

Jinbiao Chen, Zizhen Zhang, Zhiguang Cao, Yaoxin Wu, Yining Ma, Te Ye, and Jiahai Wang. Neural multi-objective combinatorial optimization with diversity enhancement. In *Advances in Neural Information Processing Systems*, volume 36, pp. 39176–39188, December 2023b.

Jinbiao Chen, Zhiguang Cao, Jiahai Wang, Yaoxin Wu, Hanzhang Qin, Zizhen Zhang, and Yue-Jiao Gong. Rethinking neural multi-objective combinatorial optimization via neat weight embedding. In *International Conference on Learning Representations*, volume 2025, pp. 48510–48529, May 2025a.

Jinbiao Chen, Jiahai Wang, Zhiguang Cao, and Yaoxin Wu. Neural multi-objective combinatorial optimization via graph-image multimodal fusion. In *International Conference on Learning Representations*, volume 2025, pp. 33964–33983, May 2025b.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002. ISSN 1941-0026. doi: 10.1109/4235.996017.

Mingfeng Fan, Yaoxin Wu, Zhiguang Cao, Wen Song, Guillaume Sartoretti, Huan Liu, and Guohua Wu. Conditional neural heuristic for multiobjective vehicle routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 36(3):4677–4689, March 2025. ISSN 2162-2388. doi: 10.1109/TNNLS.2024.3371706.

Mingfeng Fan, Jianan Zhou, Yifeng Zhang, Yaoxin Wu, Jinbiao Chen, and Guillaume Adrien Sartoretti. Preference-driven multi-objective combinatorial optimization with conditional computation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 11594–11627, June 2026. doi: 10.48550/arXiv.2506.08898.

Samira Fazlollahi, Pierre Mandel, Gwenaelle Becker, and Francois Maréchal. Methods for multi-objective investment and operating optimization of complex energy systems. *Energy*, 45(1):12–22, September 2012. ISSN 0360-5442. doi: 10.1016/j.energy.2012.02.046.

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*, 180(1):116–148, July 2007. ISSN 0377-2217. doi: 10.1016/j.ejor.2006.03.041.

Salvatore Greco, Matthias Ehrgott, and José Rui Figueira (eds.). *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 233 of *International Series in Operations Research & Management Science*. Springer New York, New York, NY, 2016. ISBN 978-1-4939-3093-7 978-1-4939-3094-4. doi: 10.1007/978-1-4939-3094-4.

Nathan Grinsztajn, Daniel Furelos-Blanco, Shikha Surana, Clément Bonnet, and Tom Barrett. Winner takes it all: Training performant rl populations for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:48485–48509, December 2023.

Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, May 2001. ISSN 0377-2217. doi: 10.1016/S0377-2217(00)00100-4.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, October 2000. ISSN 0377-2217. doi: 10.1016/S0377-2217(99)00284-2.

André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *International Conference on Learning Representations*, March 2022. doi: 10.48550/arXiv.2106.05126.

Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Transactions on Evolutionary Computation*, 19(2):264–283, April 2015. ISSN 1941-0026. doi: 10.1109/TEVC.2014.2315442.

Andrzej Jaszkiewicz. Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71, February 2002. ISSN 0377-2217. doi: 10.1016/S0377-2217(01)00104-7.

Jakob Kallestad, Ramin Hasibi, Ahmad Hemmati, and Kenneth Sörensen. A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems. *European Journal of Operational Research*, 309(1):446–468, August 2023. ISSN 0377-2217. doi: 10.1016/j.ejor.2023.01.017.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Machine Learning*. arXiv, February 2019. doi: 10.48550/arXiv.1803.08475.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21188–21198. Curran Associates, Inc., 2020.

Kaiwen Li, Tao Zhang, and Rui Wang. Deep reinforcement learning for multiobjective optimization. *IEEE Transactions on Cybernetics*, 51(6):3103–3114, June 2021. ISSN 2168-2275. doi: 10.1109/TCYB.2020.2977661.

Kaiwen Li, Tao Zhang, Rui Wang, Yuheng Wang, Yi Han, and Ling Wang. Deep reinforcement learning for combinatorial optimization: Covering salesman problems. *IEEE Transactions on Cybernetics*, 52(12):13142–13155, December 2022. ISSN 2168-2275. doi: 10.1109/TCYB.2021.3103811.

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*. arXiv, May 2022. doi: 10.48550/arXiv.2203.15386.

Si-Chen Liu, Zong-Gan Chen, Zhi-Hui Zhan, Sang-Woon Jeon, Sam Kwong, and Jun Zhang. Many-objective job-shop scheduling: A multiple populations for multiple objectives-based genetic algorithm approach. *IEEE Transactions on Cybernetics*, 53(3):1460–1474, March 2023. ISSN 2168-2275. doi: 10.1109/TCYB.2021.3102642.

Fu Luo, Xi Lin, Yaoxin Wu, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Qingfu Zhang. Boosting neural combinatorial optimization for large-scale vehicle routing problems. In *The Thirteenth International Conference on Learning Representations*, October 2024.

Shu Luo, Linxuan Zhang, and Yushun Fan. Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning. *Computers & Industrial Engineering*, 159:107489, September 2021. ISSN 0360-8352. doi: 10.1016/j.cie.2021.107489.

Thibaut Lust and Jacques Teghem. The multiobjective traveling salesman problem: A survey and a new approach. In Carlos A. Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan (eds.), *Advances in Multi-Objective Nature Inspired Computing*, pp. 119–141. Springer, Berlin, Heidelberg, 2010a. ISBN 978-3-642-11218-8. doi: 10.1007/978-3-642-11218-8_6.

Thibaut Lust and Jacques Teghem. Two-phase pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16(3):475–510, June 2010b. ISSN 1572-9397. doi: 10.1007/s10732-009-9103-9.

George Mavrotas, José Rui Figueira, and Eleftherios Siskos. Robustness analysis methodology for multi-objective combinatorial optimization problems and application to project selection. *Omega*, 52:142–155, April 2015. ISSN 0305-0483. doi: 10.1016/j.omega.2014.11.005.

Luis Paquete and Thomas Stützle. A two-phase local search for the biobjective traveling salesman problem. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Lothar Thiele, and Kalyanmoy Deb (eds.), *Evolutionary Multi-Criterion Optimization*, pp. 479–493, Berlin, Heidelberg, 2003. Springer. ISBN 978-3-540-36970-7. doi: 10.1007/3-540-36970-8_34.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *Advances in Neural Information Processing Systems*, volume 38, pp. 100092–100118, May 2025. doi: 10.48550/arXiv.2505.06708.

Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, November 2006. ISSN 1526-5447. doi: 10.1287/trsc.1050.0135.

Yinan Shao, Jerry Chun-Wei Lin, Gautam Srivastava, Dongdong Guo, Hongchun Zhang, Hu Yi, and Alireza Jolfaei. Multi-objective neural evolutionary algorithm for combinatorial optimization problems. *IEEE Transactions on Neural Networks and Learning Systems*, 34(4):2133–2143, April 2023. ISSN 2162-2388. doi: 10.1109/TNNLS.2021.3105937.

Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget (eds.), *Principles and Practice of Constraint Programming — CP98*, pp. 417–431, Berlin, Heidelberg, 1998. Springer. ISBN 978-3-540-49481-2. doi: 10.1007/3-540-49481-2_30.

Jialong Shi, Jianyong Sun, Qingfu Zhang, Haotian Zhang, and Ye Fan. Improving pareto local search using cooperative parallelism strategies for multiobjective combinatorial optimization. *IEEE Transactions on Cybernetics*, 54(4):2369–2382, April 2024. ISSN 2168-2275. doi: 10.1109/TCYB.2022.3226744.

Jiwoo Son, Zhikai Zhao, Federico Berto, Chuanbo Hua, Changhyun Kwon, and Jinkyoo Park. Neural combinatorial optimization for real-world routing. In *International Conference on Learning Representations*. arXiv, March 2025. doi: 10.48550/arXiv.2503.16159.

Ye Tian, Langchun Si, Xingyi Zhang, Ran Cheng, Cheng He, Kay Chen Tan, and Yaochu Jin. Evolutionary large-scale multi-objective optimization: A survey. *ACM Comput. Surv.*, 54(8): 174:1–174:34, October 2021. ISSN 0360-0300. doi: 10.1145/3470971.

Renato Tinós, Keld Helsgaun, and Darrell Whitley. Efficient recombination in the lin-kernighan-helsgaun traveling salesman heuristic. In Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and Darrell Whitley (eds.), *Parallel Problem Solving from Nature – PPSN XV*, pp. 95–107, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99253-2. doi: 10.1007/978-3-319-99253-2_8.

Zhenkun Wang, Shunyu Yao, Genghui Li, and Qingfu Zhang. Multiobjective combinatorial optimization using a single deep reinforcement learning model. *IEEE Transactions on Cybernetics*, 54(3):1984–1996, March 2024. ISSN 2168-2275. doi: 10.1109/TCYB.2023.3312476.

L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, February 2006. ISSN 1941-0026. doi: 10.1109/TEVC.2005.851275.

Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):5057–5069, September 2022. ISSN 2162-2388. doi: 10.1109/TNNLS.2021.3068828.

Sandra Zajac and Sandra Huber. Objectives and methods in multi-objective routing problems: A survey and classification scheme. *European Journal of Operational Research*, 290(1):1–25, April 2021. ISSN 0377-2217. doi: 10.1016/j.ejor.2020.07.005.

Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007. ISSN 1941-0026. doi: 10.1109/TEVC.2007.892759.

Zizhen Zhang, Zhiyuan Wu, Hang Zhang, and Jiahai Wang. Meta-learning-based deep reinforcement learning for multiobjective optimization problems. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):7978–7991, October 2023. ISSN 2162-2388. doi: 10.1109/TNNLS.2022.3148435.

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK report*, pp. 21 p., May 2001. doi: 10.3929/ETHZ-A-004284029.

A Detailed Description of the Studied MOCO Problems

A.1 Multi-objective Traveling Salesman Problem

Problem statement. For the multi-objective traveling salesman problem (MOTSP), an instance consists of n nodes. Each node $i \in \{1, \dots, n\}$ is associated with M groups of two-dimensional coordinates, denoted by $v_i^m \in [0,1]^2$ for objective $m \in \{1, \dots, M\}$. The Euclidean distance between nodes i and j under the m-th objective is computed as $d_{ij}^m = |v_i^m - v_j^m|_2$. A feasible solution is a tour $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ that visits each node exactly once and returns to the starting node. The cost under each objective is obtained by summing the corresponding objective-specific distances along the tour and the return edge. MOTSP aims to find a set of Pareto-optimal tours by minimizing all objective-specific tour lengths simultaneously.

Instance generation. For synthetic MOTSP instances, the coordinates of each node are independently sampled from the uniform distribution over $[0,1]^{2M}$. We consider both bi-objective and tri-objective settings, corresponding to $M = 2$ and $M = 3$, respectively.

Node features and context embedding. The input feature of node i is the concatenation of its objective-specific coordinates, i.e., $o_i = [v_i^1, v_i^2, \dots, v_i^M]$, which has dimension $2M$. At each decoding step, the context embedding is constructed by concatenating the embeddings of the first visited node and the last visited node. All previously visited nodes are masked to ensure that each node is selected exactly once.

A.2 Multi-objective Capacitated Vehicle Routing Problem

Problem statement. We consider the bi-objective capacitated vehicle routing problem (Bi-CVRP) with n customers and one depot. Each customer i has coordinate x_i and demand d_i, while the depot has zero demand. A fleet of identical vehicles with capacity Q starts and ends at the depot. Each customer is visited exactly once, and for each route r, the capacity constraint $\sum_{i \in r} d_i \leq Q$ must hold. We optimize two objectives: minimizing the total route length f_1 and minimizing the maximum route length $f_2 = \max_r \text{len}(r)$. The goal is to obtain Pareto-optimal feasible solutions.

Instance generation. For Bi-CVRP instances, depot and customer coordinates are independently sampled from the uniform distribution over $0, 1^2$. Customer demands are uniformly sampled from $1, \dots, 9$ and normalized by vehicle capacity. The capacity is set by problem size: $Q = 30$ for $20 \leq n < 40$, $Q = 40$ for $40 \leq n < 70$, and $Q = 50$ for $70 \leq n \leq 100$.

Node features and context embedding. Each customer uses its coordinate and normalized demand as features, while the depot uses its coordinate and zero demand. At each decoding step, the context consists of the last visited node and remaining capacity. Visited or infeasible customers are masked, and returning to the depot resets capacity to Q.

A.3 Multi-objective Knapsack Problem

Problem statement. We consider the bi-objective knapsack problem (Bi-KP) with n items. Each item i has a weight w_i and two objective-specific profit values. A solution is represented by a binary vector $x \in \{0,1\}^n$ subject to the capacity constraint $\sum_{i=1}^n w_i x_i \leq C$. The objective is to simultaneously maximize the two total profits and identify Pareto-optimal feasible solutions.

Instance generation. For Bi-KP instances, the weight and the two profit values of each item are independently sampled from the uniform distribution over $[0,1]$. Following the standard neural MOKP setting, the knapsack capacity is set to 12.5 for $50 \leq n < 100$ and 25 for $100 \leq n \leq 200$.

Node features and context embedding. Each item is treated as a node, and its input feature is composed of its weight and two profit values. At each decoding step, the context embedding is constructed from the graph embedding and the remaining knapsack capacity. Previously selected items are masked, and items whose weights exceed the remaining capacity are also masked.

Algorithm 1 REINFORCE-based training

1: **Input:** Instance distribution $\mathcal{P}_s$, preference distribution P_λ, training steps E, batch size B, number of rollouts K, scalarized cost $C(\pi \mid \boldsymbol{\lambda}, s)$
2: **Output:** Trained policy network p_θ
3: Initialize model parameters θ
4: **for** $e = 1$ to E **do**
5: **for** $b = 1$ to B **do**
6: Sample an instance $s_b \sim \mathcal{P}_s$ and a preference vector $\boldsymbol{\lambda}_b \sim P_\lambda$
7: **for** $k = 1$ to K **do**
8: Generate a solution $\pi_b^k \sim p_\theta(\cdot \mid \boldsymbol{\lambda}_b, s_b)$
9: Compute the reward $R_b^k \leftarrow -C(\pi_b^k \mid \boldsymbol{\lambda}_b, s_b)$
10: **end for**
11: Compute the shared baseline $B_b \leftarrow \frac{1}{K} \sum_{k=1}^{K} R_b^k$
12: **end for**
13: Calculate the REINFORCE gradient $\nabla_\theta \mathcal{L}_{\mathrm{RL}}$
14: Update $\theta \leftarrow \mathrm{Adam}(\theta, \nabla_\theta \mathcal{L}_{\mathrm{RL}})$
15: **end for**

B Training Algorithms

This section provides the training procedures for PGDA-enhanced neural MOCO solvers. Depending on the backbone architecture, PGDA is optimized using either REINFORCE-based training or pairwise preference-based training.

B.1 REINFORCE-based Training

For the basic PGDA model and non-POCCO backbones, we train a unified preference-conditioned policy using REINFORCE. Algorithm 1 summarizes the procedure. After initializing the policy parameters in Line 3, each iteration samples a mini-batch of instances and preference vectors in Lines 5–6. For each instance–preference pair, the policy samples multiple solution trajectories in Lines 7–8, and their rewards are computed as the negative scalarized costs in Line 9. To reduce gradient variance, Line 11 computes a shared baseline by averaging the trajectory rewards. The REINFORCE gradient is calculated in Line 13, followed by an Adam update in Line 14.

B.2 Pairwise Preference-based Training

For POCCO-based variants, including POCCO-C-PGDA and POCCO-W-PGDA, we follow the pairwise preference-based learning strategy. Algorithm 2 summarizes the detailed procedure. After initializing the policy network in Line 3, each training iteration samples a mini-batch of instances and preference vectors in Lines 5–6. Given each instance-preference pair, the current policy generates two candidate solutions in Line 7. The candidates are then ranked by their scalarized costs in Lines 8–12. The lower-cost solution is regarded as the winner, and the other is regarded as the loser. Their normalized log-likelihood scores are computed in Line 13. The pairwise loss in Line 15 encourages the winner to receive a higher likelihood score than the loser. The parameters are updated using Adam in Line 16.

C Hypervolume Indicator

We use hypervolume (HV) While et al. (2006) as the main metric for evaluating approximate Pareto fronts, as it captures both convergence and diversity. Given an approximate Pareto front $\mathcal{P}$ and a reference point r^p, HV is defined as

$$\mathrm{HV}(\mathcal{P}, r^p) = \Lambda\left(\bigcup_{p \in \mathcal{P}} \{q \mid r^p \prec q \prec p\}\right), \tag{14}$$

where $\Lambda(\cdot)$ denotes the Lebesgue measure, i.e., the hypervolume of the dominated region in the objective space. $\mathcal{P}$ denotes the approximate PF produced by a method, and p is an objective vector

Algorithm 2 Pairwise preference-based training

Input: Instance distribution $\mathcal{P}_s$, preference distribution P_λ, training steps E, batch size B, sharpness parameter β, scalarized cost $C(\pi \mid \boldsymbol{\lambda}, s)$
Output: Trained policy network p_θ
Initialize model parameters θ
for $e = 1$ to E **do**
  for $b = 1$ to B **do**
    Sample an instance $\int_b \sim \mathcal{P}_s$ and a preference vector $\boldsymbol{\lambda}_b \sim P_\lambda$
    Sample two solutions $\pi_b^a, \pi_b^b \sim p_\theta(\cdot \mid \boldsymbol{\lambda}_b, s_b)$
    if $C(\pi_b^a \mid \boldsymbol{\lambda}_b, s_b) \leq C(\pi_b^b \mid \boldsymbol{\lambda}_b, s_b)$ **then**
      $\pi_b^w \leftarrow \pi_b^a, \pi_b^l \leftarrow \pi_b^b$
    else
      $\pi_b^w \leftarrow \pi_b^b, \pi_b^l \leftarrow \pi_b^a$
    end if
    Compute normalized scores $r_\theta(\pi_b^w \mid \boldsymbol{\lambda}_b, s_b)$ and $r_\theta(\pi_b^l \mid \boldsymbol{\lambda}_b, s_b)$
  end for
  Calculate the pairwise gradient $\nabla_\theta \mathcal{L}_{\text{pair}}(\theta)$
  Update $\theta \leftarrow \text{Adam}(\theta, \nabla_\theta \mathcal{L}_{\text{pair}})$
end for

Table 4: Reference and ideal points used for HV calculation.

Problem	Size	Reference point	Ideal point
Bi-TSP	20	$(20, 20)$	$(0, 0)$
	50	$(35, 35)$	$(0, 0)$
	100	$(65, 65)$	$(0, 0)$
	150	$(85, 85)$	$(0, 0)$
	200	$(115, 115)$	$(0, 0)$
Bi-CVRP	20	$(30, 4)$	$(0, 0)$
	50	$(45, 4)$	$(0, 0)$
	100	$(80, 4)$	$(0, 0)$
Bi-KP	50	$(5, 5)$	$(30, 30)$
	100	$(20, 20)$	$(50, 50)$
	200	$(30, 30)$	$(75, 75)$
Tri-TSP	20	$(20, 20, 20)$	$(0, 0, 0)$
	50	$(35, 35, 35)$	$(0, 0, 0)$
	100	$(65, 65, 65)$	$(0, 0, 0)$

in $\mathcal{P}$. The point q represents any objective vector in the objective-space region bounded by p and the reference point r^p. The notation $a \prec b$ means that a dominates b. Therefore, the set $\{q \mid r^p \prec q \prec p\}$ denotes the hyper-rectangle dominated by p and bounded by r^p.

Since the scales of objective values differ across problem types and sizes, we report the normalized HV, computed as $\overline{\text{HV}}(\mathcal{P}) = \text{HV}(\mathcal{P}, r^p) / \prod_{m=1}^{M} |r_m^p - z_m|$, where z denotes the ideal point and M is the number of objectives. All methods use the same reference point r^p and ideal point z for each benchmark. The settings are summarized in Table 4.

D Instance Augmentation

Instance augmentation is applied during inference to further improve solution quality for routing-based problems. The key idea is that Euclidean routing instances can be transformed into equivalent instances without changing the optimal tour or route structure. For a two-dimensional coordinate (x, y), we use the following eight transformations:(x, y), (y, x), $(x, 1-y)$, $(y, 1-x)$, $(1-x, y)$, $(1-y, x)$, $(1-x, 1-y)$, $(1-y, 1-x)$. For Bi-CVRP, each node has one coordinate group, and thus each instance has 8 augmented variants. For MOTSP, each objective is associated with an independent coordinate group. Therefore, the transformations are applied independently to each objective-specific coordinate group, yielding $8^2 = 64$ augmented variants for Bi-TSP and $8^3 = 512$

augmented variants for Tri-TSP. The policy solves the augmented instances, maps the generated solutions back to the original instance, and keeps the best or nondominated candidates for final HV evaluation. No coordinate-based instance augmentation is used for Bi-KP.

E Theoretical Interpretation of PGDA

This section provides a theoretical interpretation of PGDA. Recent gated attention studies show that applying a sigmoid gate to the attention output can improve attention modules and stabilize training Qiu et al. (2025). PGDA follows this general gated-attention principle, but extends it from input-conditioned gating to preference-conditioned and output-aware gating. Specifically, PGDA generates the gate from the pre-attention representation X, the preference representation Pref, and the post-attention output Y. We do not claim that PGDA guarantees global convergence. Instead, we show that PGDA generalizes gated attention, introduces residual-aware output modulation, and provides a preference-conditioned gradient filtering mechanism.

E.1 PGDA as a Generalization of Gated Attention

Let X denote the pre-attention representation and Y denote the output of a standard attention:

$$Y = \mathrm{Attn}(X). \tag{15}$$

A typical gated attention module generates a gate from the input representation and uses it to modulate the attention output Qiu et al. (2025):

$$G_X = \sigma(\phi_X(X)), \qquad \tilde{Y} = G_X \odot Y, \tag{16}$$

where $\phi_X(\cdot)$ is the gate generator and $\odot$ denotes element-wise multiplication.

PGDA extends this formulation by incorporating the preference representation and the attention output itself:

$$G = \sigma(\phi_g([X; \mathrm{Pref}; Y])), \qquad \hat{Y} = G \odot Y. \tag{17}$$

Proposition 1. PGDA strictly generalizes input-conditioned gated attention.

Proof. Consider the PGDA gate generator:

$$G = \sigma(\phi_g([X; \mathrm{Pref}; Y])). \tag{18}$$

If the parameters associated with Pref and Y in $\phi_g(\cdot)$ are set to zero, the gate reduces to a function of X only:

$$G = \sigma(\phi_X(X)). \tag{19}$$

Therefore, the gated-attention form in Eq. (16) is a special case of PGDA. PGDA preserves the original gated-attention mechanism while further allowing the gate to depend on preference information and post-attention content.

This proposition clarifies the relationship between PGDA and existing gated attention. PGDA does not discard the original input-conditioned gate. Instead, it augments it with two additional sources of information: Pref for objective conditioning and Y for output-aware modulation.

E.2 Residual-aware and Output-aware Modulation

PGDA uses both the pre-attention representation X and the post-attention representation Y to generate the gate. This design is related in spirit to residual learning He et al. (2016), because X provides a reference for the original representation while Y represents the attention-transformed representation. However, PGDA is not an additive residual block of the form $X + Y$. Instead, it uses X to decide how the attention output Y should be selectively preserved or suppressed.

To see this more clearly, consider the first linear layer of the gate generator:

$$Z = W_X X + W_P \mathrm{Pref} + W_Y Y + b. \tag{20}$$

Since Y can be decomposed into the original representation and the attention-induced change,

$$Y = X + (Y - X), \tag{21}$$

Eq. (20) can be rewritten as

$$Z = (W_X + W_Y)X + W_Y(Y - X) + W_P \mathrm{Pref} + b. \tag{22}$$

Proposition 2. Using both X and Y allows PGDA to model both the original representation and the attention-induced residual change.

Proof. Eq. (7) shows that a gate generated from $X; \mathrm{Pref}; Y$ can be equivalently written as a function of X, Pref, and $Y - X$. Here, X is the pre-attention reference, while $Y - X$ captures the attention-induced change. Using only Y cannot explicitly separate the original information from this change, whereas using only X ignores the attention output. Thus, combining X and Y enables residual-aware and output-aware modulation.

This interpretation explains the roles of X and Y in PGDA. The attention module first produces a weighted representation Y, while the gate uses both X and Y to determine whether the attended features are consistent with, complementary to, or deviated from the original representation. Thus, PGDA can be viewed as a residual-aware feature selector over the attention output.

E.3 Preference-conditioned Gradient Filtering

The most important difference between PGDA and standard gated attention is the preference input. In neural MOCO, different preference vectors correspond to different scalarized subproblems. For a preference vector $\boldsymbol{\lambda}$, a scalarized cost can be written as

$$C(\pi, \boldsymbol{\lambda}) = \sum_{m=1}^{M} \lambda_m f_m(\pi). \tag{23}$$

Thus, changing $\boldsymbol{\lambda}$ changes the optimization direction of the policy. PGDA incorporates this preference information into the gate:

$$D_{\boldsymbol{\lambda}} = \mathrm{diag}(G(X, \mathrm{Pref}, Y)), \qquad \hat{Y} = D_{\boldsymbol{\lambda}} Y. \tag{24}$$

Proposition 3. PGDA provides a bounded and preference-conditioned gradient filtering.

Proof. Since G is produced by a sigmoid function, each gate value satisfies $0 < G_l < 1$. Therefore,

$$\|\hat{Y}\|_2 = \|D_{\boldsymbol{\lambda}} Y\|_2 \leq \|Y\|_2. \tag{25}$$

This shows that the direct modulation path does not amplify the norm of the attention output.

For the backward pass, the direct gradient path satisfies

$$\frac{\partial \mathcal{L}_{\boldsymbol{\lambda}}}{\partial Y} = D_{\boldsymbol{\lambda}} \frac{\partial \mathcal{L}_{\boldsymbol{\lambda}}}{\partial \hat{Y}}, \tag{26}$$

because $D_{\boldsymbol{\lambda}}$ is diagonal. Therefore, PGDA acts as a preference-conditioned diagonal gradient filter.

This property is useful for unified preference-conditioned learning. Consider two preferences $\boldsymbol{\lambda}^a$ and $\boldsymbol{\lambda}^b$, with downstream gradients u^a and u^b. Their directly modulated gradient inner product is

$$\left\langle D_{\boldsymbol{\lambda}^a} u^a, D_{\boldsymbol{\lambda}^b} u^b \right\rangle = \sum_l g_l^a g_l^b u_l^a u_l^b. \tag{27}$$

If $u_l^a u_l^b < 0$, then the two preferences have conflicting gradient directions on dimension l. The factor $g_l^a g_l^b$ can reduce the contribution of this conflicting dimension when the learned gates assign small weights to it. Therefore, PGDA does not guarantee the elimination of gradient conflict, but it provides a preference-conditioned mechanism to attenuate conflicting feature dimensions among different scalarized subproblems.

PGDA also affects preference-conditioned search. If the decoder computes the logit of action a as

$$s_a = w_a^\top \hat{y}, \tag{28}$$

then the margin between two candidate actions a and b is

$$s_a - s_b = (w_a - w_b)^\top (g \odot y) = \sum_l g_l (w_{a,l} - w_{b,l}) y_l. \tag{29}$$

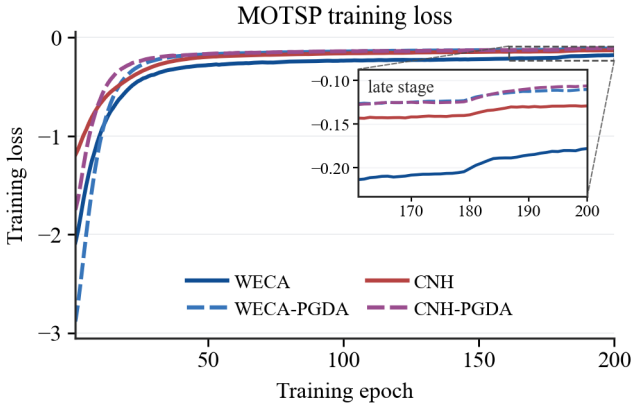

Figure 5: Training loss curves on MOTSP. The inset zooms in on the late training stage. PGDA-enhanced variants reach a more stable surrogate-loss plateau than their corresponding backbones.

Thus, the gate directly changes the action margin. By conditioning the gate on Pref, PGDA can emphasize feature dimensions that are more relevant to the current scalarized objective and suppress less useful dimensions. This provides a theoretical explanation for why PGDA can improve preference-specific search behavior.

Figure 5 further illustrates the training dynamics on MOTSP. Compared with their corresponding backbones, the PGDA-enhanced variants approach a stable surrogate-loss plateau more rapidly and exhibit smoother late-stage behavior. In particular, WE-CA-PGDA stabilizes around -0.10 in the late stage, whereas WE-CA remains around -0.20. A similar trend can also be observed for CNH and CNH-PGDA, where the PGDA-enhanced variant reaches a slightly higher and smoother plateau. Since the REINFORCE loss is a surrogate objective affected by rewards, baselines, and log-probabilities, the curve should not be interpreted as a proof of global convergence. Rather, it provides empirical evidence that PGDA stabilizes the policy-gradient training process, which is consistent with the bounded modulation and preference-conditioned gradient filtering analysis above.

F Detailed Results on Benchmark Instances

F.1 KroAB Benchmark Instances

Table 5 reports detailed results on KroAB instances with 100, 150, and 200 nodes. For compactness, -C" and -W" denote POCCO-C and POCCO-W, respectively, with the same convention for PGDA-enhanced and augmented variants. The comparison includes non-learnable, multi-model, single-model, and POCCO-based solvers with their augmented versions. Overall, WS-LKH achieves the best HV on all three instances but requires substantially longer running time than neural solvers. Among learnable methods, PGDA-enhanced variants perform strongly. In particular, POCCO-W-PGDA-aug obtains the second-best HV on all three instances while maintaining much shorter inference time than classical non-learnable solvers. These results further show that PGDA improves preference-conditioned neural solvers beyond randomly generated test sets.

F.2 Large-scale Bi-TSP Benchmark Instances

Table 6 reports results on large-scale Bi-TSP instances with 150 and 200 nodes. WS-LKH achieves the best HV and Gap but requires much longer computation. Among neural methods, PGDA-enhanced variants remain competitive. In particular, POCCO-W-PGDA achieves the second-best HV and Gap on both Bi-TSP150 and Bi-TSP200, showing that PGDA improves preference-conditioned search on larger instances.

Table 5: Detailed results on KroAB benchmark instances.

Method	KroAB100			KroAB150			KroAB200		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
WS-LKH	**0.7022**	**-0.36%**	2.3m	**0.7017**	**-1.05%**	4.0m	**0.7430**	**-2.07%**	5.6m
PGDA	0.6969	0.40%	**10s**	0.6927	0.25%	**15s**	0.7315	-0.49%	**29s**
POCCO-C	0.6965	0.45%	30s	0.6925	0.27%	40s	0.7302	-0.31%	50s
POCCO-W	0.6981	0.23%	20s	0.6946	-0.03%	31s	0.7345	-0.90%	40s
-C-PGDA	0.6983	0.20%	33s	0.6937	0.10%	46s	0.7320	-0.56%	60s
-W-PGDA	0.6991	0.08%	31s	0.6956	-0.17%	45s	0.7346	-0.92%	60s
PGDA-Aug	0.6998	-0.02%	**10s**	0.6963	-0.27%	25s	0.7352	-1.00%	45s
POCCO-C-Aug	0.6999	-0.03%	32s	0.6959	-0.21%	48s	0.7334	-0.75%	1.1m
POCCO-W-Aug	0.7006	-0.13%	22s	0.6976	-0.46%	39s	0.7369	-1.23%	1.1m
-C-PGDA-Aug	0.7007	-0.20%	36s	0.6971	-0.39%	60s	0.7348	-0.94%	1.3m
-W-PGDA-Aug	0.7012	-0.22%	33s	0.6982	-0.55%	50s	0.7379	-1.37%	1.3m

Table 6: Detailed results on large-scale Bi-TSP benchmark instances.

Method	Bi-TSP150			Bi-TSP200		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
WS-LKH	**0.7149**	**-1.71%**	13h	**0.7490**	**-2.50%**	22h
MOGLS	0.6768	3.71%	22h	0.7114	2.65%	38h
PPLS/D-C	0.6784	3.48%	21h	0.7106	2.76%	32h
MOEA/D	0.6809	3.13%	2.4h	0.7139	2.30%	2.7h
NSGA-II	0.6659	5.26%	6.8h	0.7045	3.59%	6.9h
MDRL	0.6922	1.52%	**40s**	0.7251	0.77%	**1.4m**
EMNH	0.6930	1.41%	**40s**	0.7260	0.65%	**1.4m**
PMOCO	0.6910	1.69%	50s	0.7231	1.04%	1.5m
CNH	0.6985	0.62%	1.1m	0.7292	0.21%	1.9m
WE-CA	0.7008	0.30%	57s	0.7346	-0.53%	1.6m
PGDA	0.7018	0.15%	70s	0.7356	-0.67%	2m
-C	0.7011	0.25%	1.5m	0.7333	-0.35%	2.5m
-W	0.7033	-0.06%	1.4m	0.7371	-0.87%	2.4m
-C-PGDA	0.7030	-0.02%	2.1m	0.7347	-0.54%	3.5m
-W-PGDA	0.7042	-0.19%	1.9m	0.7384	-1.05%	3.1m

F.3 Tri-objective TSP Benchmark Instances

Table 7 reports the detailed results on Tri-TSP benchmark instances with 20, 50, and 100 nodes. Compared with bi-objective cases, Tri-TSP is more challenging because the policy needs to approximate a higher-dimensional Pareto front. The results show that PGDA-enhanced variants remain competitive under different numbers of sampled preference vectors. In particular, POCCO-W-PGDA achieves strong HV and Gap performance on larger preference sets, indicating that PGDA can help the unified policy adapt to more diverse trade-off directions in tri-objective settings.

G Generality of PGDA Beyond POCCO-based Models

Figure 6 examines whether the effectiveness of PGDA depends on POCCO-based training. We report the HV gain of each PGDA-enhanced model over its corresponding backbone on Bi-TSP instances. As shown in Figure 6(a), PGDA consistently improves the standalone CNH and WE-CA solvers, with particularly clear gains on larger problem scales. This suggests that PGDA can directly enhance preference-conditioned representation learning without relying on POCCO-based pairwise training. Figure 6(b) further shows that PGDA also improves POCCO-based variants in most cases. The gains over POCCO-C and POCCO-W indicate that PGDA provides complementary benefits to POCCO-style training. Overall, these results demonstrate that PGDA is a general preference-aware modulation module rather than a component specifically tailored to POCCO-based models.

Table 7: Selected results on Tri-TSP benchmark instances.

N	Method	Tri-TSP20			Tri-TSP50			Tri-TSP100		
		HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
105	WS-LKH	**0.4712**	**0.00%**	12m	**0.4440**	**-0.20%**	1.9h	**0.5076**	**-0.79%**	6.6h
	PGDA	0.4709	0.06%	**3s**	0.4400	0.70%	**7s**	0.4986	1.00%	**20s**
	-C	0.4704	0.17%	8s	0.4393	0.86%	17s	0.4985	1.02%	28s
	-W	0.4710	0.04%	6s	0.4397	0.77%	13s	0.4985	1.02%	23s
	-C-PGDA	0.4707	0.11%	8s	0.4400	0.70%	15s	0.4994	0.84%	30s
	-W-PGDA	**0.4712**	**0.00%**	9s	0.4414	0.39%	15s	0.5013	0.46%	39s
1035	PGDA	0.4746	-0.72%	**28s**	0.4540	-2.46%	**1.2m**	0.5181	-2.87%	**3.0m**
	-C	0.4751	-0.83%	48s	0.4548	-2.64%	1.7m	0.5188	-3.01%	8.1m
	-W	**0.4752**	**-0.85%**	67s	0.4550	-2.68%	2.5m	0.5193	-3.11%	8.2m
	-C-PGDA	0.4750	-0.81%	52s	0.4553	-2.75%	2.0m	0.5199	-3.23%	9.1m
	-W-PGDA	0.4748	-0.76%	54s	**0.4554**	**-2.77%**	2.0m	**0.5206**	**-3.37%**	6.8m
10011	PGDA	0.4755	-0.91%	**4.7m**	0.4574	-3.22%	**12m**	0.5235	-3.95%	**29m**
	-C	**0.4763**	**-1.08%**	12m	0.4594	-3.68%	25m	0.5251	-4.26%	34m
	-W	**0.4763**	**-1.08%**	11m	0.4595	-3.70%	25m	0.5254	-4.32%	36m
	-C-PGDA	**0.4763**	**-1.08%**	13m	**0.4598**	**-3.77%**	25m	0.5261	-4.46%	35m
	-W-PGDA	0.4759	-1.00%	9m	0.4594	-3.68%	20m	**0.5264**	**-4.52%**	30m

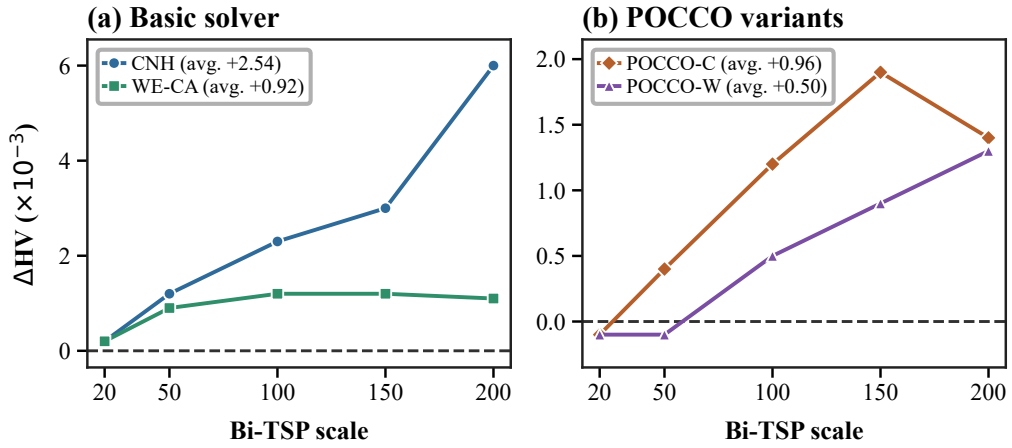

Figure 6: HV gains of PGDA over the corresponding backbones on Bi-TSP, in units of 10^{-3}. Parentheses show average gains across scales.

H Sensitivity Analysis of PGDA Design

To investigate the robustness of PGDA, we conduct a sensitivity analysis on three design factors: gate depth, hidden dimension, and insertion position. Experiments are performed on Bi-TSP100/150/200 and augmented Kro100/150/200 instances. The default POCCO-W-PGDA configuration used in the main experiments is denoted as "Ours", while POCCO-W without PGDA serves as a reference baseline. Figure 7 reports the HV difference between each variant and Ours; values closer to zero indicate performance closer to the default design.

Gate depth. The top row of Figure 7 compares different gate depths. Both $L = 0$ and $L = 2$ underperform the default setting across Bi-TSP and augmented Kro instances. A shallow gate may lack sufficient nonlinear capacity to model interactions among the original representation, preference embedding, and attention output, whereas a deeper gate adds complexity without improving performance. These results suggest that the default gate depth provides a suitable balance between modeling capacity and optimization stability.

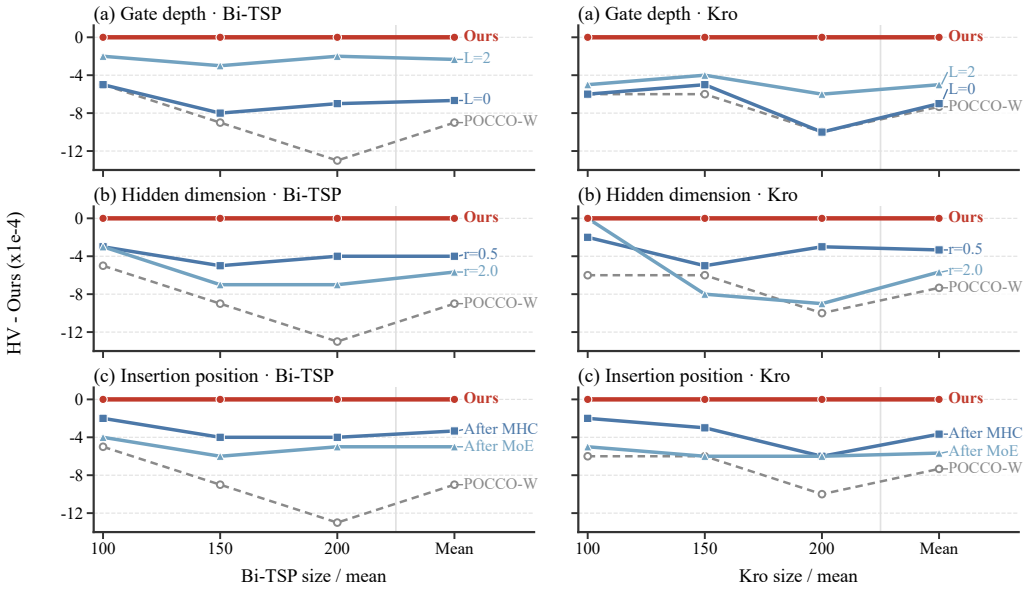

Figure 7: Sensitivity analysis of PGDA design on Bi-TSP100/150/200 and augmented Kro100/150/200 instances. The y-axis reports the HV difference between each variant and Ours.

Hidden dimension. The middle row compares the default hidden dimension with $r = 0.5$ and $r = 2.0$, where r controls the relative hidden dimension of the gate generator. Both reducing and enlarging the dimension generally lower HV. A smaller dimension may limit fine-grained preference-aware modulation, while a larger dimension introduces additional capacity without consistent gains. The default setting therefore provides a reasonable trade-off between representation capacity and generalization.

Insertion position. The bottom row compares the default insertion position with alternatives that place the gate after the multi-head combination module or the MoE module. Both alternatives reduce HV across Bi-TSP and augmented Kro instances, indicating that later insertion does not provide equivalent benefits. The default design modulates the attention output before subsequent feature transformations, allowing preference information to influence the representation flow earlier and more effectively. Delaying this modulation weakens its effect on later feature processing and preference-guided adaptation.

I Summary of Decomposition-based Neural MOCO Solvers

Table 8 summarizes representative decomposition-based neural MOCO solvers in terms of learning strategy, solving paradigm, and parameter count. In contrast to one-to-one methods, which train separate models for different preferences or subproblems, one-to-many solvers learn a unified preference-conditioned model. PGDA follows the one-to-many paradigm and introduces preference-aware gated modulation into the neural solver. Although integrating PGDA increases the parameter count relative to some lightweight backbones, the resulting model remains substantially smaller than one-to-one neural solvers such as MDRL and EMNH. When integrated with POCCO-based backbones, PGDA also maintains a moderate parameter scale while improving preference-conditioned optimization performance.

Table 8: Summary of the decomposition-based neural MOCOP solvers.

Method	Learning method	Paradigm	#Parameters
MDRL	Meta learning + RL	one-to-one	133.37M
EMNH	Meta learning + RL	one-to-one	133.37M
PMOCO	RL	one-to-many	1.50M
CNH	RL	one-to-many	1.63M
WE-CA	RL	one-to-many	1.47M
POCCO-C	Preference learning	one-to-many	2.16M
POCCO-W	Preference learning	one-to-many	2.00M
PGDA	RL	one-to-many	2.71M
POCCO-C-PGDA	Preference learning	one-to-many	3.60M
POCCO-W-PGDA	Preference learning	one-to-many	3.38M

J LICENSES FOR EXISTING ASSETS

The assets used in this work are summarized in Table 9. All listed assets are either open-source or available for academic research under their respective licenses or usage terms. The source code of this work has been released under the MIT License to facilitate reproducibility and future research. The relevant references include LKH Helsgaun (2000), MDRL Zhang et al. (2023), EMNH Chen et al. (2023a), PMOCO Lin et al. (2022), CNH Fan et al. (2025), WE-CA Chen et al. (2025a), POCCO Fan et al. (2026), and the datasets introduced by Chen et al. Chen et al. (2025a). The corresponding license information and original sources are provided in the table for completeness. This information supports transparent reuse and facilitates verification of the experimental setup.

Table 9: Used assets, licenses, and their usage.

Type	**Asset**	**License**	**Usage**
Code	LKH Helsgaun (2000)	Available for academic use	Evaluation
	MDRL Zhang et al. (2023)	MIT License	Evaluation
	EMNH Chen et al. (2023a)	MIT License	Evaluation
	PMOCO Lin et al. (2022)	MIT License	Evaluation
	CNH Fan et al. (2025)	MIT License	Revision
	WE-CA Chen et al. (2025a)	MIT License	Revision
	POCCO Fan et al. (2026)	MIT License	Revision
Datasets	Chen et al. Chen et al. (2025a)	MIT License	Evaluation

AI USE STATEMENT

Generative AI tools were used solely to improve the clarity, grammar, and readability of the manuscript. All technical content, experimental design, analysis, results, and conclusions were produced and verified by the authors. The use of AI did not involve generating experimental data, deriving scientific claims, or making research decisions.