Preference-Driven Multi-Objective Combinatorial Optimization with Conditional Computation

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Recent deep reinforcement learning methods have achieved remarkable success in solving multi-objective combinatorial optimization problems (MOCOPs) by decomposing them into multiple subproblems, each associated with a specific weight vector. However, these methods typically treat all subproblems equally and solve them using a single model, hindering the effective exploration of the solution space and thus leading to suboptimal performance. To overcome the limitation, we propose POCCO, a novel plug-and-play framework that enables adaptive selection of model structures for subproblems, which are subsequently optimized based on preference signals rather than explicit reward values. Specifically, we design a conditional computation block that routes subproblems to specialized neural architectures. Moreover, we propose a preference-driven optimization algorithm that learns pairwise preferences between winning and losing solutions. We evaluate the efficacy and versatility of POCCO by applying it to two state-of-the-art neural methods for MOCOPs. Experimental results across two classic MOCOP benchmarks demonstrate its significant superiority and strong generalization.

1 Introduction

Multi-objective combinatorial optimization problems (MOCOPs) require optimizing multiple conflicting objectives in a discrete decision space. Due to their broad applications in manufacturing [1], logistics [16], and scheduling [12], they have attracted significant attention from computer science and operations research communities. Unlike single-objective problems, MOCOPs seek a set of Pareto optimal solutions that capture trade-offs among objectives such as cost, makespan, and environmental impact. Their NP-hard nature makes exact methods impractical for large instances [9, 11], leading to widespread use of heuristics. However, conventional heuristics often entail costly iterative searches and require domain knowledge and fine-tuning, limiting their scalability and generalization.

Recent neural methods have achieved notable success in solving SOCOPs [2, 26, 6, 15, 17, 20, 34, 38] by learning decision policies in a data-driven manner. Building on this progress, researchers have extended neural approaches to MOCOPs, which offer advantages such as avoiding heuristic design, enabling GPU acceleration, and adapting to diverse problem variants. Most methods decompose MOCOPs into scalarized subproblems, each defined by a weight vector, and apply deep reinforcement learning (DRL) to approximate the Pareto front. Early works train separate models per subproblem using transfer or meta-learning [21, 37], but suffer from high computational cost and poor generalization to unseen weights. PMOCO [23] addresses this by using a weight-conditioned hypernetwork to modulate model parameters within a single model, yet struggles with diverse weights. Recent methods like CNH [10] and WE-CA [4] improve generalization by embedding weight vectors into problem representations, achieving state-of-the-art performance across varying problem sizes.

Current SOTA methods typically rely on a single neural network with limited capacity to handle all subproblems, which overcomplicates the learning task and results in suboptimal performance. A straightforward solution to ease training and promote effective representation learning across subproblems is to increase the model capacity. However, determining *how much* additional capacity to allocate and *where* to introduce it within the architecture remains an open challenge. On the other hand, neural methods often adopt REINFORCE [33] as the training algorithm, relying solely on scalarized objective values as reward signals to guide policy updates. Given its on-policy nature, REINFORCE suffers from high gradient variance and lacks structured mechanisms for effective exploration [19]. These issues are exacerbated in MOCOP settings, where the vast combinatorial action space makes efficient exploration particularly difficult, ultimately hindering policy performance.

To address these issues, we propose POCCO (Preference-driven multi-objective combinatorial Optimization with Conditional COmputation), a plug-and-play framework that augments neural MOCOP methods with two complementary mechanisms. First, POCCO introduces a *conditional computation block* into the decoder, where a sparse gating network dynamically routes each subproblem through either a selected subset of feed-forward (FF) experts or a parameter-free identity (ID) expert. This design enables subproblems to adaptively select computation routes (i.e., model structures) based on their context, efficiently scaling model capacity and facilitating more effective representation learning. Second, POCCO replaces raw scalarized rewards with *pairwise preference learning*. For each subproblem, the policy samples two trajectories, identifies the better one as the winner, and maximizes a Bradley–Terry (BT) likelihood based on the difference in their average log-likelihoods. Such comparative feedback guides the search toward policies that generate increasingly preferred solutions, enabling exploration of the most promising regions of the search space and more efficient convergence to higher-quality solutions.

Our contributions are summarized as follows: 1) Conceptually, we address two fundamental limitations of existing approaches for solving MOCOPs: limited exploration within the vast solution space and the reliance on a single, capacity-limited model, which can lead to inefficient learning and suboptimal performance. 2) Technically, we propose a conditional computation block that dynamically routes subproblems to tailored neural architectures. Additionally, we develop a preference-driven algorithm leveraging implicit rewards derived from pairwise preference signals between winning and losing solutions, modeled using the BT framework. 3) Experimentally, we demonstrate the effectiveness and versatility of POCCO on classical MOCOP benchmarks using two SOTA neural methods. Extensive results show that POCCO not only outperforms all baseline methods but also exhibits superior generalization across diverse problem sizes.

2 Methodology

2.1 Overview

POCCO is a learning-based framework that trains a portfolio of policies to solve a set of scalarized subproblems $\{(\mathcal{G}, \lambda_i)\}_{i=1}^{N}$, obtained by decomposing an MOCOP instance. Instead of forcing a single policy to handle all subproblems which often yields bland and suboptimal behavior, POCCO promotes specialization: each policy is encouraged to focus on a subset of subproblems, yielding a diverse policy ensemble. Such diversity is known to enhance multi-task optimization [30] by expanding the exploration space and ultimately improving solution quality. Technically, we achieve this diversity by activating different subsets of model parameters through a CCO block, enabling distinct computational paths to emerge for different subproblems. Moreover, POCCO should encourage each policy to thoroughly explore the combinatorial solution space during training for reducing suboptimality. To achieve this, we replace raw rewards with preference signals. For each subproblem $(\mathcal{G}, \lambda_i)$, we construct a set of winning–losing solution pairs $\{(\pi^{w,j}, \pi^{l,j})\}_{j=1}^{K}$. Training then proceeds by maximizing the likelihood of the winning solutions while minimizing that of the losing ones, following a BT-style objective. This preference-driven training encourages the learned policy to explore the most promising regions of the search space, leading to more efficient convergence toward higher-quality solutions. Notably, POCCO is a generic, plug-and-play framework that can seamlessly involve different neural solvers for MOCOPs. We demonstrate this by augmenting two SOTA methods, CNH [10] and WE-CA [4], in Section 3.

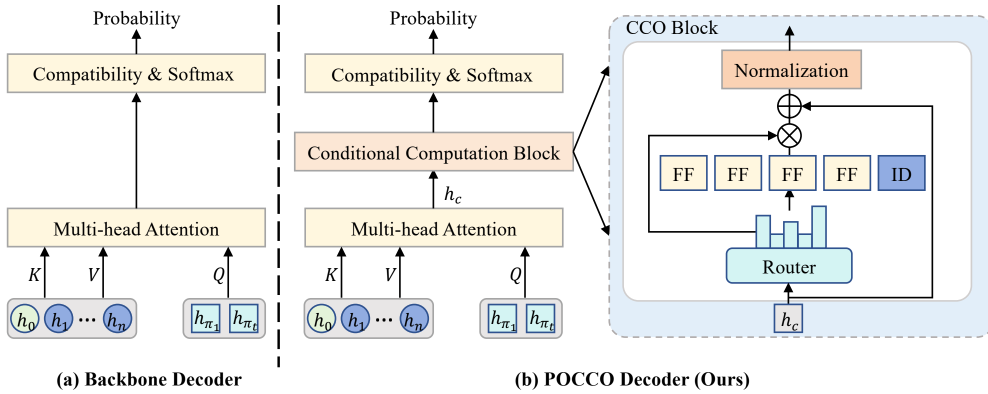


Figure 1: Decoder structures of backbone and POCCO.

2.2 Conditional Computational Block

Most approaches employ a Transformer-based architecture, where the encoder generates joint node embeddings $\{h_i\}_{i=0}^n$ that capture the interaction between the instance $\mathcal{G}$ and the weight vector λ_i, and the decoder produces candidate solutions conditioned on these embeddings. We propose a CCO block to increase the model capacity and promote policy diversity across subproblems. To maintain efficiency, we integrate the CCO block solely into the decoder of the backbone model. This design enables the generation of multiple diverse solutions through a single, computationally expensive encoder pass, offering a favorable trade-off between empirical performance and computational cost.

As illustrated in Fig. 1, the CCO block comprises multiple FF experts and a single ID expert. We insert this block between the multi-head attention (MHA) layer and the compatibility layer in the decoder. Given a batch of MHA outputs $\{h_c^b\}_{b=1}^B$, the CCO block dynamically routes each context vector h_c^b from the corresponding subproblem through either the FF or ID experts, forming distinct computation paths that function as different policies. The ID expert allows the model to bypass the FF computation, promoting architectural sparsity and specialization [13]. Consequently, the CCO block facilitates the learning of dedicated, weight-specific policies tailored to individual subproblems.

Formally, a CCO block consists of: 1) m FF experts $\{E_1, E_2, \ldots, E_m\}$ with independent trainable parameters; 2) a parameter-free identity expert E_{m+1}; 3) a router, implemented as a gating network G parameterized by W_G, which determines how the inputs $\{h_c^b\}_{b=1}^B$ are routed to the experts; and 4) a skip connection followed by an instance normalization (IN) layer. Given a single context vector h_c^b, let $G(h_c^b) \in \mathbb{R}^{m+1}$ denote the output of the gating network, which represents the expert selection probabilities, and let $E_j(h_c^b)$ denote the output of the j-th expert. The output of the CCO block is:

$$\mathrm{CCO}(h_c^b) = \mathrm{IN}\left(\sum_{j=1}^{m+1} G(h_c^b)_j E_j(h_c^b) + h_c^b\right). \tag{1}$$

The sparse vector $G(h_c^b)$ activates only a small subset of experts, either parameterized FF experts or the parameter-free ID expert, thereby enabling diverse computation paths while reducing computational overhead. A typical implementation uses a $\mathrm{Top}\,k$ operator that retains the k largest logits and masks the rest with $-\infty$. In this case, the gating network output is: $G(h_c^b) = \mathrm{Softmax}\big(\mathrm{Top}\,k(h_c^b \cdot W_G)\big)$.

Our proposed CCO block aligns with the principles of recent advances in efficiently scaling Transformer-based models along both width [28] and depth [27]. In specific, it combines of a mixture-of-experts (MoE) layer, implemented using multiple FF experts to widen the network, with a mixture-of-depths (MoD) layer, realized through an ID expert that allows inputs to skip computation. Within the CCO block, each subproblem is adaptively routed to only a small subset of experts, granting the model the expressiveness of a significantly wider network while preserving computational efficiency. As demonstrated in Section 3, this joint design achieves a better capacity–efficiency trade-off than scaling either dimension in isolation.

2.3 Preference-driven MOCO

To mitigate the exploration inefficiencies inherent in REINFORCE algorithms, we optimize *relative preferences* [25] instead of absolute objective values.

Generating preference pairs. For each scalarized subproblem $(\mathcal{G}, \lambda_i)$ in the training batch, the policy p_θ samples two candidate solutions, π^w and π^l. We denote $\pi^w \lessdot \pi^l$ if π^w is preferred over π^l, as determined by the ordering of their scalarized objective values. This evaluator ranks the two solutions and designates the better one as the winning solution π^w and the other as the losing solution π^l. A binary preference label y is then assigned, where $y = 1$ if $\pi^w \lessdot \pi^l$, and $y = 0$ otherwise. This label serves as the supervision signal required for the preference-driven MOCO.

Defining an implicit reward. Distinct from DRL training paradigms that rely on raw objective values, POCCO treats the average log-likelihood of a solution as an implicit reward f_θ, directly relating preferences between solutions to their policy probabilities. This reward is inherently normalized by sequence length $|\pi|$, thereby mitigating length bias between winning and losing solution pairs.

$$f_\theta(\pi|\mathcal{G}, \lambda_i) = \frac{1}{|\pi|} \log\, p_\theta(\pi|\mathcal{G}, \lambda_i) = \frac{1}{|\pi|} \sum_{t=1}^{|\pi|} \log\, p_\theta(\pi_t|\pi_{<t}, \mathcal{G}, \lambda_i). \tag{2}$$

Learning from pairwise comparisons. We formulate preference learning (PL) as a probabilistic binary classification problem through the BT model. Specifically, the BT model is a pairwise preference framework that uses a function $g_\theta(\cdot)$ to map reward differences into preference probabilities. It assigns each solution a strength proportional to its implicit reward (defined in Eq. (2)) and predicts the probability g_θ that the winning solution outranks the losing one:

$$g_\theta\big(\pi^w \lessdot \pi^\ell \mid \mathcal{G}, \lambda_i\big) = \sigma\big(\beta\,[\, f_\theta(\pi^w \mid \mathcal{G}, \lambda_i) - f_\theta(\pi^\ell \mid \mathcal{G}, \lambda_i)\,]\big), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function, and $\beta > 0$ is a fixed temperature that controls the sharpness with which the model distinguishes between unequal rewards. We maximize the likelihood of the collected preferences, yielding the following loss function:

$$\mathcal{L}(\theta|p_\theta, \mathcal{G}, \lambda_i, \pi^w, \pi^l) = -y\, \log\, \sigma\big(\beta[\frac{\log\, p_\theta(\pi^w|\mathcal{G}, \lambda_i)}{|\pi^w|} - \frac{\log\, p_\theta(\pi^l|\mathcal{G}, \lambda_i)}{|\pi^l|}]\big). \tag{4}$$

In practice, we collect multiple (π^w, π^ℓ) pairs per update, sum their losses, and backpropagate through p_θ. By maximizing the log-likelihood of $g_\theta\big(\pi^w \lessdot \pi^\ell \mid \mathcal{G}, \lambda_i\big)$, the model is encouraged to assign higher probabilities to preferred solutions π^w over less preferred ones π^l.

3 Experiments

3.1 Experimental settings

Training. We conduct extensive experiments to evaluate the effectiveness of the proposed POCCO across two representative MOCOPs: the multi-objective traveling salesman problem (MOTSP)[24] and the multi-objective capacitated vehicle routing problem (MOCVRP)[35]. In MOTSP, the objective is to determine a tour that visits each node exactly once while minimizing multiple path lengths, each calculated using a distinct set of coordinates corresponding to different objectives. In MOCVRP, a fleet of capacity-constrained vehicles must serve all customer nodes and return to a central depot, with the goals of minimizing the total travel distance and the length of the longest individual route. We evaluate POCCO on three commonly used problem sizes: $n = 20/50/100$.

Hyperparameters. We implement POCCO on top of two SOTA neural MOCO methods, CNH [10] and WE-CA [4], resulting in POCCO-C and POCCO-W, respectively. Most hyperparameters are aligned with those used in the original CNH and WE-CA implementations. Both models are trained for 200 epochs, with each epoch processing 100,000 randomly sampled instances and a batch size of $B = 64$. We use the Adam optimizer [18] with a learning rate of 3×10^{-4} and a weight decay of 10^{-6}. We generate the $N = 101$ weight vectors for decomposition using the method proposed in [7].

Baselines. We compare POCCO with a broad range of baseline methods across three categories, all employing weighted-sum (WS) scalarization to ensure fair comparison: (1) Single-model neural

Table 1: Performance on BiTSP and MOCVRP Instances

	Bi-TSP20			Bi-TSP50			Bi-TSP100		
Method	HV	Gap	Time	HV	Gap	Time	HV	Gap	Time
WS-LKH	0.6270	0.00%	10m	0.6415	0.05%	1.8h	**0.7090**	-0.17%	6h
MOEA/D	0.6241	0.46%	1.7h	0.6316	1.59%	1.8h	0.6899	2.53%	2.2h
NSGA-II	0.6258	0.19%	6.0h	0.6120	4.64%	6.1h	0.6692	5.45%	6.9h
MOGLS	**0.6279**	-0.14%	1.6h	0.6330	1.37%	3.7h	0.6854	3.16%	11h
PPLS/D-C	0.6256	0.22%	26m	0.6282	2.12%	2.8h	0.6844	3.31%	11h
DRL-MOA	0.6257	0.21%	6s	0.6360	0.90%	9s	0.6970	1.53%	16s
MDRL	0.6271	-0.02%	5s	0.6364	0.84%	8s	0.6969	1.54%	14s
EMNH	0.6271	-0.02%	5s	0.6364	0.84%	8s	0.6969	1.54%	15s
PMOCO	0.6259	0.18%	6s	0.6351	1.04%	12s	0.6957	1.71%	26s
CNH	0.6270	0.00%	13s	0.6387	0.48%	16s	0.7019	0.83%	33s
POCCO-C	0.6275	-0.08%	14s	0.6409	0.14%	20s	0.7047	0.44%	42s
WE-CA	0.6270	0.00%	6s	0.6392	0.41%	9s	0.7034	0.62%	18s
POCCO-W	0.6275	-0.08%	7s	0.6411	0.11%	14s	0.7055	0.32%	36s
CNH-Aug	0.6271	-0.02%	1.3m	0.6410	0.12%	3.9m	0.7054	0.34%	12m
POCCO-C-Aug	0.6270	0.00%	2.2m	0.6416	0.03%	4.0m	0.7071	0.10%	14m
WE-CA-Aug	0.6271	-0.02%	1.3m	0.6413	0.08%	3.6m	0.7066	0.17%	12m
POCCO-W-Aug	0.6270	0.00%	2.2m	**0.6418**	0.00%	4.0m	0.7078	0.00%	14m

	MOCVRP20			MOCVRP50			MOCVRP100		
Method	HV	Gap	Time	HV	Gap	Time	HV	Gap	Time
MOEA/D	0.4255	1.07%	2.3h	0.4000	2.63%	2.9h	0.3953	3.33%	5.0h
NSGA-II	0.4275	0.60%	6.4h	0.3896	5.16%	8.8h	0.3620	11.47%	9.4h
MOGLS	0.4278	0.53%	9.0h	0.3984	3.02%	20h	0.3875	5.23%	72h
PPLS/D-C	0.4287	0.33%	1.6h	0.4007	2.46%	9.7h	0.3946	3.50%	38h
DRL-MOA	0.4287	0.33%	8s	0.4076	0.78%	12s	0.4055	0.83%	21s
MDRL	0.4291	0.23%	6s	0.4082	0.63%	13s	0.4056	0.81%	22s
EMNH	0.4299	0.05%	7s	0.4098	0.24%	12s	0.4072	0.42%	22s
PMOCO	0.4267	0.79%	6s	0.4036	1.75%	12s	0.3913	4.30%	22s
CNH	0.4287	0.33%	11s	0.4087	0.51%	15s	0.4065	0.59%	25s
POCCO-C	0.4294	0.16%	16s	0.4101	0.17%	25s	0.4079	0.24%	53s
WE-CA	0.4290	0.26%	7s	0.4089	0.46%	10s	0.4068	0.51%	21s
POCCO-W	0.4294	0.16%	8s	0.4102	0.15%	17s	0.4084	0.12%	46s
CNH-Aug	0.4299	0.05%	21s	0.4101	0.17%	45s	0.4077	0.29%	1.9m
POCCO-C-Aug	**0.4302**	-0.02%	31s	**0.4108**	0.00%	1.4m	0.4086	0.07%	2.4m
WE-CA-Aug	0.4300	0.02%	15s	0.4103	0.12%	36s	0.4081	0.20%	1.8m
POCCO-W-Aug	0.4301	0.00%	24s	**0.4108**	0.00%	1.2m	**0.4089**	0.00%	2.3m

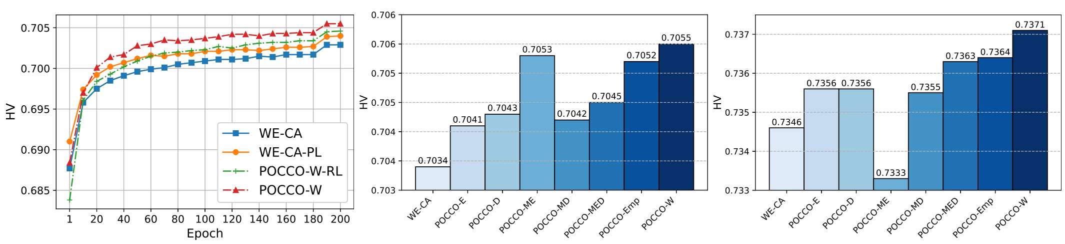
MOCO approaches: This includes **PMOCO** [23], and recent SOTA methods **CNH** [10], and **WE-CA** [4]. Both CNH and WE-CA are unified model trained across problem size $n \in \{20, 21, \cdots, 100\}$. (2) Multi-model neural MOCO approaches: This category covers methods like **DRL-MOA** [21], **MDRL** [37], and **EMNH** [5]. (3) Non-learnable approaches, including classical MOEAs and other problem-specific heuristics: **MOEA/D** [36] and **NSGA-II** [8], each run for 4,000 iterations, serve as representative decomposition-based and dominance-based MOEAs, respectively. Finally, **WS-LKH** combines weighted-sum scalarization with the powerful LKH solver [14, 29] for solving MOTSP.

Inference. We evaluate all methods using three metrics: average hypervolume (HV) [31], average gap, and total runtime per instance set. HV is a widely used indicator in multi-objective optimization that reflects both the convergence and diversity of the solution set. A higher HV indicates better performance. To ensure consistency, HV values are normalized to the range $[0, 1]$ using the same reference point for all methods. The gap is defined as the relative difference between a method's HV and the HV of POCCO-W. Methods with the "-Aug" suffix apply instance augmentation [23] to further improve performance. To evaluate statistical significance, we use the Wilcoxon rank-sum test [32] at a 1% significance level. The best result and others that are not significantly worse are marked in bold, while the second-best and statistically similar results are underlined. All experiments are implemented in Python and conducted on a machine with NVIDIA Ampere A100-80GB GPUs and an AMD EPYC 7742 CPU. The code and dataset will be released publicly upon acceptance.

3.2 Experimental results

Comparison analysis. The comparison results are presented in Table 1. POCCO-W consistently achieves superior performance over WE-CA across all benchmark scenarios, establishing itself as the new SOTA results among neural MOCOP solvers. Similarly, POCCO-C outperforms CNH in every case. Both variants also surpass their augmentation-based counterparts, WE-CA-Aug and

CNH-Aug, on Bi-TSP20 and MOCVRP100, highlighting POCCO's enhanced ability to explore the solution space and approximate high-quality Pareto fronts. When further combined with instance augmentation, POCCO demonstrates additional performance gains. Please note that POCCO with instance augmentation yields lower HV values compared with its non-augmented counterpart on Bi-TSP20. This is because decomposition-based methods focus on optimizing individual subproblems rather than ensuring overall solution diversity. While augmentation improves solution quality for specific subproblems, it may reduce the number of non-dominated solutions, resulting in a smaller HV. Compared with multi-model approaches that require training or fine-tuning separate models for each subproblem, POCCO delivers superior results while maintaining a single shared model. Notably, POCCO achieves better results on Bi-TSP50 than WS-LKH, a setting where previous neural solvers have consistently failed. In terms of efficiency, POCCO significantly reduces computational time. For example, POCCO-W-Aug solves Bi-TSP100 in only 14 minutes, while WE-LKH requires about 6.0 hours, with POCCO delivering comparable solution quality.



(a) Validation curves (b) Bi-TSP100 (c) Bi-TSP200

Figure 2: Ablation study:(a) validates the effectiveness of PL; (b) and (c) verify the effects of different CCO block variants.

Effectiveness of the PL. We assess the training efficiency of the PL by comparing it to REINFORCE on the WE-CA and POCCO-W models using the Bi-TSP100 dataset. As shown by the validation curves in Fig. 2a, PL achieves faster convergence despite identical network architectures. Notably, for WE-CA, training with PL for 100 epochs reaches performance comparable to 200 epochs of REINFORCE. Similar improvements are observed for POCCO-W. These results demonstrate that PL effectively accelerates training process and achieves better performance with fewer training epochs.

Effectiveness of the CCO block. To evaluate the impact of the CCO block's structure and placement, we compare POCCO-W with WE-CA and several POCCO variants: POCCO-E (CCO inserted in the encoder replacing the FF layer), POCCO-D (CCO replacing the final linear layer of MHA in the decoder, using MLP experts), POCCO-ME (replacing CCO with a standard MoE layer), POCCO-MD (replacing CCO with three MoD layers), POCCO-MED (using MoE in the encoder and MoD in place of CCO), and POCCO-Emp (replacing the identity expert in CCO with an empty expert). As shown in Fig. 2b, all variants outperform WE-CA on the in-distribution Bi-TSP100, with POCCO-W, POCCO-ME, and POCCO-Emp achieving the most notable gains. On the out-of-distribution Bi-TSP200 in Fig. 2c, only POCCO-W and POCCO-Emp maintain strong performance, while POCCO-ME performs worst, even underperforming WE-CA. These results highlight the importance of both the structure and placement of the CCO block for achieving strong generalization across in- and out-of-distribution settings.

4 Conclusion

This paper presents POCCO, a plug-and-play framework tailored for MOCOPs, which adaptively routes subproblems through different model structures and leverages PL for more effective training. POCCO is integrated into two SOTA neural solvers, and extensive experiments demonstrate its effectiveness. Ablation studies further highlight the necessity of both CCO block and PL, and reveal the critical impact of the design and placement of CCO block. We acknowledge certain limitations, such as the limited capability to address real-world MOCOPs with complex constraints or large problem sizes. Addressing these challenges may require constraint-handling mechanisms [3] or divide-and-conquer [22] strategies, which we leave for future work.

References

[1] Ehsan Ahmadi, Mostafa Zandieh, Mojtaba Farrokh, and Seyed Mohammad Emami. A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers & operations research*, 73:56–66, 2016.

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

[3] Jieyi Bi, Yining Ma, Jianan Zhou, Wen Song, Zhiguang Cao, Yaoxin Wu, and Jie Zhang. Learning to handle complex constraints for vehicle routing problems. In *Advances in Neural Information Processing Systems*, volume 37, pages 93479–93509, 2024.

[4] 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 *The Thirteenth International Conference on Learning Representations*, 2025.

[5] Jinbiao Chen, Jiahai Wang, Zizhen Zhang, Zhiguang Cao, Te Ye, and Siyuan Chen. Efficient meta neural heuristic for multi-objective combinatorial optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[7] I Das and JE Dennis. Normal-boundary intersection: A new method for generating pareto-optimal points in multieriteria optimization problems. *SIAM J. Optimiz*, 1996.

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

[9] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[10] 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*, 2024.

[11] Kostas Florios and George Mavrotas. Generation of the exact pareto set in multi-objective traveling salesman and set covering problems. *Applied Mathematics and Computation*, 237:1–19, 2014.

[12] Keivan Ghoseiri, Ferenc Szidarovszky, and Mohammad Jawad Asgharpour. A multi-objective train scheduling model and solution. *Transportation research part B: Methodological*, 38(10):927–952, 2004.

[13] Jiayi Han, Liang Du, Hongwei Du, Xiangguo Zhou, Yiwen Wu, Weibo Zheng, and Donghong Han. Slim: Let llm learn more and forget less with soft lora and identity mixture. In *North American Chapter of the Association for Computational Linguistics Annual Conference*, 2025.

[14] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European journal of operational research*, 126(1):106–130, 2000.

[15] André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *International Conference on Learning Representations*, 2021.

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

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

[18] Diederik P Kingma. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[19] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[20] Jingwen Li, Yining Ma, Zhiguang Cao, Yaoxin Wu, Wen Song, Jie Zhang, and Yeow Meng Chee. Learning feature embedding refiner for solving vehicle routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

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

[22] Sirui Li, Zhongxia Yan, and Cathy Wu. Learning to delegate for large-scale vehicle routing. In *Advances in Neural Information Processing Systems*, volume 34, pages 26198–26211, 2021.

[23] Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022.

[24] Thibaut Lust and Jacques Teghem. The multiobjective traveling salesman problem: A survey and a new approach. In *Advances in Multi-Objective Nature Inspired Computing*, pages 119–141. Springer, 2010.

[25] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

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

[27] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

[28] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[29] Renato Tinós, Keld Helsgaun, and Darrell Whitley. Efficient recombination in the lin-kernighan-helsgaun traveling salesman heuristic. In *Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part I 15*, pages 95–107. Springer, 2018.

[30] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *Advances in Neural Information Processing Systems*, 37:124420–124450, 2024.

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

[32] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*, pages 196–202. Springer, 1992.

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

[34] 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*, 2021.

[35] 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, 2021.

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

[37] 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*, 2022.

[38] Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Towards omni-generalizable neural methods for vehicle routing problems. In *International Conference on Machine Learning*, 2023.