SUPL2I: STANDALONE LEARNING TO IMPROVE FOR MULTI-OBJECTIVE COMBINATORIAL OPTIMISATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural methods have shown strong performance on single-objective combinatorial optimisation problems (SOCOPs), through both learning to construct solutions from scratch (L2C) and learning to improve complete solutions with local moves (L2I). However, existing neural methods for multi-objective combinatorial optimisation problems (MOCOPs) are predominantly based on the L2C paradigm. The few existing L2I methods for MOCOPs are integrated into external search frameworks (e.g., an MOEA or local search). When designing a standalone L2I method for MOCOPs, there are several issues worth exploring: 1) different objectives may relate to different features (e.g., node coordinates or costs in multi-objective TSP), so a single representation of all these features may not reflect the effect of local moves on each objective separately; 2) unlike L2C, L2I additionally needs to consider node positions, so where to inject the preference is non-trivial; 3) in SOCOPs, a local move typically makes a solution either better or worse, whereas in MOCOPs, a local move can lead to an incomparable solution (i.e., one that is nondominated to the original solution).

In this paper, we look at these three issues and propose a standalone preference-conditioned L2I method unified across problem sizes. Specifically, the proposed method 1) builds one representation per objective from its related node features, with a shared encoder, and scores local moves under each objective separately; 2) keeps node positions preference-free while injecting the preference into both the node embeddings and the fusion of the per-objective scores; and 3) considers the nondominance relation (i.e., only accepting superior and incomparable solutions) in training and inference. Moreover, we introduce a positional encoding that captures the cyclic and hierarchical structure of routing solutions and transfers across problem sizes. Experiments show that our method substantially outperforms state-of-the-art neural and conventional methods in both in-distribution and cross-size generalisation settings. Notably, in the in-distribution setting, it even outperforms the strong classical solver LKH with weighted-sum scalarisation while requiring less time.

1 INTRODUCTION

Multi-objective combinatorial optimisation problems (MOCOPs) require the simultaneous optimisation of multiple conflicting objectives over a finite discrete decision space (Ehrgott, 2005). Unlike single-objective combinatorial optimisation problems (SOCOPs), where there usually exists a single optimal solution, MOCOPs have a set of Pareto optimal solutions that represent different trade-offs. MOCOPs exist in many fields, ranging from software engineering (Chen & Li, 2023) to machine learning (Sukthanker et al., 2025; Zhao et al., 2025; Khan & Bidgoli, 2026), and from logistics (Zajac & Huber, 2021) to molecular design (Jain et al., 2023). As exact methods can become computationally intractable as problem size grows (Ehrgott et al., 2016), conventional methods such as multi-objective evolutionary algorithms (MOEAs) are widely used to approximate the Pareto front.

Recently, neural combinatorial optimisation (NCO) has received increasing attention due to its strong performance on SOCOPs (Bengio et al., 2021; Berto et al., 2025). NCO can be broadly divided into three categories: *learning-to-construct* (L2C), *learning-to-improve* (L2I), and hybrid methods. L2C methods build a solution from scratch (Kool et al., 2019; Kwon et al., 2020; Luo et al., 2026),

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.

whereas L2I methods refine a complete solution with local operators (Chen & Tian, 2019; Lu et al., 2019; Wu et al., 2021; Ma et al., 2021; Hottung & Tierney, 2022; Ma et al., 2023). Hybrid methods combine classical or neural constructive solvers with learned components that decide, for example, which part of the solution to re-optimise (Li et al., 2021; Hottung et al., 2025; Ouyang et al., 2026).

Despite rapid development of L2I for SOCOPs, most neural methods follow the L2C paradigm for MOCOPs (Li et al., 2020; Lin et al., 2022; Fan et al., 2025; Chen et al., 2023a; 2025). To our knowledge, the very few existing studies that adopt L2I for MOCOPs integrate learning within a conventional heuristic (e.g., an MOEA or local search), such as PI (Zheng et al., 2023) and WRPN (Ye et al., 2025). These methods rely on an external search framework, with the model used to control or enhance specific components rather than to automate the design of the overall search process.

Designing a standalone L2I method is non-trivial. Several questions need to be answered: 1) how should a solution be represented under multiple conflicting objectives? 2) where in the model should the preference be injected? and 3) which candidates are worth exploring?

First, different objectives may relate to different features, e.g., node coordinates or costs per objective in the multi-objective TSP (MOTSP) (Lust & Teghem, 2010). A single representation of all these features may not reflect the effect of local moves on each objective separately.

Second, where to inject the preference is not straightforward. In L2C, the preference conditions either the node embeddings (Fan et al., 2025; Chen et al., 2025) or the decoder (Lin et al., 2022). L2I, however, also needs to represent the node positions in the current solution (Hua et al., 2026), which do not depend on the preference. Existing L2I methods for MOCOPs nevertheless inject the preference only into the input embeddings, i.e., the node embeddings (Zheng et al., 2023) or even both the node and positional embeddings (Ye et al., 2025), and never at the decision stage.

Third, in SOCOPs a local move usually makes a solution either better or worse, whereas in MOCOPs, the new solution can also be incomparable to the old one (i.e., the two solutions are mutually nondominated); such nondominated solutions are often worth exploring (as it promotes the spread). Existing L2I methods for MOCOPs (Zheng et al., 2023; Ye et al., 2025) follow the always-accept rule of single-objective L2I (Ma et al., 2021). Since L2I models are trained on trajectories of only a few hundred steps (Wu et al., 2021; Ma et al., 2021), training may be more efficient if the search is focused only on nondominated solutions and superior solutions (i.e. dominating ones).

In this paper, we explore the above three issues and propose SUPL2I, a Standalone Unified Preference-conditioned L2I method for MOCOPs, in which a single model serves all preferences and problem sizes without any external search framework. Our contributions are summarised as follows:

- We propose a standalone L2I method for MOCOPs: a single policy approximating an entire Pareto front by iteratively refining complete solutions. SUPL2I maintains one embedding stream per objective with a shared encoder, keeps node positions preference-free, and injects the preference into both node embeddings and the fusion of per-objective scores.
- We introduce a size-aligned cyclic rotary positional encoding that captures the circularity and hierarchy of routing solutions and transfers across problem sizes, at no cost in parameters.
- We present a nondominance-based acceptance that only considers superior solutions (i.e., dominating solutions) and nondominated ones to make training and inference more efficient.
- We show that SUPL2I substantially outperforms 12 state-of-the-art neural methods and conventional methods in both in-distribution and cross-size generalisation settings. In the in-distribution setting, it even outperforms the strong classical solver LKH with weighted-sum scalarisation while requiring less time .

2 Related Work

Traditional methods for MOCOPs. Traditional methods for MOCOPs can be divided into exact and heuristic approaches. Exact methods can find the entire Pareto set, but become impractical as problem size grows (Lust & Teghem, 2010; Reiter & Gutjahr, 2012). Instead, heuristic methods approximate the Pareto front with a set of well-distributed non-dominated solutions (Ehrgott & Gandibleux, 2004). Among them, MOEAs have long been the go-to solution, e.g., NSGA-II (Deb et al., 2002) and MOEA/D (Zhang & Li, 2007). In addition, local search methods (Liang & Li,

2026) have been found effective in solving MOCOPs (Li et al., 2024), served as standalone solvers or refinement components in hybrid evolutionary frameworks (Jaszkiewicz, 2002; Shi et al., 2022).

L2C methods for MOCOPs. Neural constructive methods for MOCOPs primarily decompose an MOCOP into a set of scalarised subproblems, and solve them via either *multi-model* or *single-model* approaches. The *multi-model* methods learn a separate model for each subproblem (Wu et al., 2019; Li et al., 2020; Shao et al., 2021; Zhang et al., 2021). Meta-learning was later introduced for fast adaptation across subproblems (Zhang et al., 2022b; Chen et al., 2023a). In contrast, *single-model* methods use one preference-conditioned model to solve all subproblems (Lin et al., 2022). Such studies include improving diversity (Chen et al., 2023b), model capacity (Fan et al., 2026), and preference–instance fusion (Fan et al., 2025; Chen et al., 2025; Wu et al., 2026).

L2I methods for MOCOPs. L2I has been extensively studied for SOCOPs (Ma et al., 2021; 2022; Hottung & Tierney, 2022; Kim et al., 2023), but remains rarely explored for MOCOPs, where learning is integrated into an external search framework (e.g., Pareto local search or MOEAs) to control neighbourhood exploration. Within Pareto local search, learned components decide when to stop scanning the neighbourhood of a selected solution and which solution to select (Zhang et al., 2022a; 2025). Within MOEAs, a preference-conditioned policy refines a selected solution via local moves for only a limited number of steps (four or five in PI (Zheng et al., 2023), fifty to a hundred in WRPN (Ye et al., 2025)), leaving the choice of which solutions to improve and keep to the framework.

3 PRELIMINARIES

3.1 MULTI-OBJECTIVE COMBINATORIAL OPTIMISATION

Formally, a minimisation MOCOP with m objectives can be defined as:

$$\min_{\mathbf{x}\in\mathcal{X}} \ \mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\big), \tag{1}$$

where $\mathbf{x}$ denotes a feasible solution in a discrete, finite decision space $\mathcal{X}$, and $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$ maps $\mathbf{x}$ to an objective vector $\mathbf{z} = (z_1, \ldots, z_m)$. In multi-objective routing problems, a solution $\mathbf{x}$ is represented as an ordered sequence of nodes. The image of $\mathcal{X}$ under $\mathbf{f}$ defines the objective space:

$$Z = \{\mathbf{z} \in \mathbb{R}^m \mid \mathbf{z} = \mathbf{f}(\mathbf{x}),\ \mathbf{x} \in \mathcal{X}\}. \tag{2}$$

Definition 1 (Pareto dominance). *Given two solutions $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, $\mathbf{x}$ weakly Pareto dominates $\mathbf{x}'$, denoted by $\mathbf{x} \preceq \mathbf{x}'$, if $f_i(\mathbf{x}) \le f_i(\mathbf{x}')$ for all $i \in \{1, \ldots, m\}$. Moreover, $\mathbf{x}$ Pareto dominates $\mathbf{x}'$, denoted by $\mathbf{x} \prec \mathbf{x}'$, if $\mathbf{x} \preceq \mathbf{x}'$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}')$ for at least one $j \in \{1, \ldots, m\}$.*

Definition 2 (Pareto optimality). *A solution $\mathbf{x}^* \in \mathcal{X}$ is Pareto optimal if there exists no $\mathbf{x} \in \mathcal{X}$ such that $\mathbf{x} \prec \mathbf{x}^*$. The set of all Pareto optimal solutions is called the Pareto set (PS), and its image in the objective space is the Pareto front (PF).*

Decomposition Methods. Decomposition transforms an MOCOP into a set of scalarised subproblems associated with different weight vectors (i.e., preferences) (Zhang & Li, 2007). Each preference $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m) \in \Delta^{m-1}$, where $\lambda_i \ge 0$ and $\sum_{i=1}^m \lambda_i = 1$, specifies a trade-off among the objectives and defines a scalarised subproblem.

A common choice is the weighted sum (WS) scalarisation, which combines the objectives linearly, $g^{\mathrm{WS}}_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x})) = \sum_{i=1}^m \lambda_i f_i(\mathbf{x})$. WS scalarisation cannot find non-supported solutions, i.e., those not lying on the convex hull of the Pareto front (Miettinen, 1999). Alternatively, the weighted Tchebycheff scalarisation minimises the maximum weighted deviation from an ideal point $\mathbf{z}^*$, $g^{\mathrm{Tche}}_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x})) = \max_{1\le i\le m} \{\lambda_i \, |f_i(\mathbf{x}) - z_i^*|\}$, where $z_i^* \le \min_{\mathbf{x}\in\mathcal{X}} f_i(\mathbf{x})$ is an ideal value for objective f_i. It can find non-supported solutions, but could yield weakly Pareto optimal solutions (Helfrich et al., 2023).

A simple yet effective way is to combine the two (Steuer & Choo, 1983), applied here to objectives normalised by a reference point $\mathbf{z}^{\mathrm{ref}}$, with the ideal point at the origin since all objectives are non-negative:

$$g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x})) = \max_{1\le i\le m} \big\{\lambda_i \bar{f}_i(\mathbf{x})\big\} + \rho \sum_{i=1}^m \lambda_i \bar{f}_i(\mathbf{x}), \qquad \bar{f}_i(\mathbf{x}) = f_i(\mathbf{x})/z_i^{\mathrm{ref}}, \tag{3}$$

where $\rho = 0.05$ and $\mathbf{z}^{\mathrm{ref}}$ is the reference point (see Appendix E for details).

3.2 POSITIONAL ENCODING IN L2I METHODS

The original Transformer adds absolute positional embeddings to token embeddings (Vaswani et al., 2017), which early Transformer-based L2I methods adopt directly (Wu et al., 2021). Subsequently, DACT (Ma et al., 2021) introduces cyclic PE (CPE). More recently, Hua et al. (2026) identify hierarchical structure, in addition to circularity, as a desirable property for vehicle routing solutions.

To incorporate relative positional information directly into attention, rotary positional encoding (RoPE) (Su et al., 2024) applies position-dependent rotations to query and key vectors. Specifically, for one attention head with even query/key dimension d_h, let $\boldsymbol{q}_i, \boldsymbol{k}_j \in \mathbb{R}^{d_h}$ denote the query and key vectors before rotation, at sequence positions p_i and p_j. The attention logit is

$$\alpha_{ij} = \frac{\big(\boldsymbol{R}(p_i)\boldsymbol{q}_i\big)^\top \big(\boldsymbol{R}(p_j)\boldsymbol{k}_j\big)}{\sqrt{d_h}} = \frac{\boldsymbol{q}_i^\top \boldsymbol{R}(p_j - p_i)\boldsymbol{k}_j}{\sqrt{d_h}}, \tag{4}$$

where $\boldsymbol{R}(p)$ rotates each pair of feature dimensions by an angle proportional to p, with a fixed frequency per pair. The rotation term depends on the relative offset $p_j - p_i$.

4 PROBLEM FORMULATION

A solution $\mathbf{x}$ is a sequence of N nodes, and p_i denotes the position of node i. Given a preference $\boldsymbol{\lambda} \in \Delta^{m-1}$, the subproblem is to minimise the scalarised objective $g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x}))$ of Eq. (3). A preference-conditioned policy refines an initial complete solution $\mathbf{x}_0$ by iteratively proposing local moves, which we formulate as the following MDP.

State. $s_t = (\boldsymbol{v}_t, \boldsymbol{p}_t, a_{t-1})$ consists of the node features $\boldsymbol{v}_t$ and node positions $\boldsymbol{p}_t$ of the current solution $\mathbf{x}_t$, and the previous move a_{t-1}, used to mask moves that repeat or undo it.
Action. $a_t = (\omega_t, i_t, j_t)$ applies the local operator $\omega_t \in \Omega$ to the node pair (i_t, j_t) (Appendix A).
Transition. The candidate $\tilde{\mathbf{x}}_{t+1} = \mathcal{T}_{\omega_t}(\mathbf{x}_t, i_t, j_t)$ is accepted unless the current one dominates it:

$$\mathbf{x}_{t+1} \leftarrow \begin{cases} \mathbf{x}_t, & \mathbf{x}_t \prec \tilde{\mathbf{x}}_{t+1}, \\ \tilde{\mathbf{x}}_{t+1}, & \text{otherwise.} \end{cases} \tag{5}$$

For MOCVRP, infeasible solutions are allowed along a trajectory but penalised (Gendreau et al., 1994; Vidal, 2022), and dominance is evaluated on the penalised objectives (Appendix A.2).
Reward. Let $\mathbf{x}_t^*$ be the best solution up to step t under $g_{\boldsymbol{\lambda}}$, with $\mathbf{x}_0^* = \mathbf{x}_0$. The reward is the improvement of the best scalarised value, $r_t = \max\{0,\ g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x}_t^*)) - g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x}_{t+1}))\}$ (Ma et al., 2021).
Policy. A single stochastic policy $\pi_\theta(a_t \mid s_t, \boldsymbol{\lambda})$, shared across preferences and problem sizes, samples a_t at every step of training and inference.

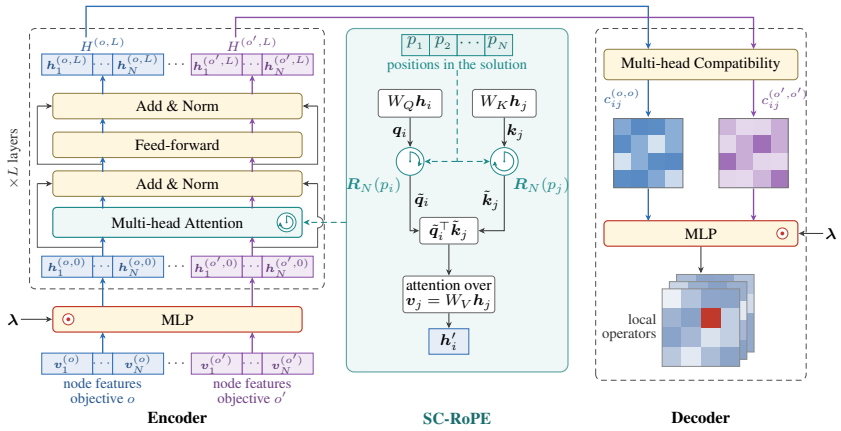


Figure 1: Model architecture of SUPL2I.

5 METHODOLOGY

5.1 OBJECTIVE-WISE REPRESENTATION

As aforementioned, SUPL2I maintains one stream of node embeddings per objective, scores a candidate local move under each objective, and combines the streams only through these scores. For objective $o \in \{1, \dots, m\}$, an input MLP maps the features $\boldsymbol{v}_i^{(o)} \in \mathbb{R}^{Z_o}$ of node i to a d-dimensional embedding, which the preference then modulates feature-wise (Perez et al., 2018):

$$\boldsymbol{h}_i^{(o,0)} = (\mathbf{1} + \boldsymbol{\gamma}_0) \odot \mathrm{MLP}_o\left(\boldsymbol{v}_i^{(o)}\right) + \boldsymbol{\beta}_0, \qquad (\boldsymbol{\gamma}_0, \boldsymbol{\beta}_0) = W_{\mathrm{in}}\boldsymbol{\lambda}^{(o)} + \boldsymbol{b}_{\mathrm{in}}, \tag{6}$$

where $W_{\mathrm{in}} \in \mathbb{R}^{2d \times m}$ and $\boldsymbol{b}_{\mathrm{in}} \in \mathbb{R}^{2d}$ are shared across objectives, $\boldsymbol{\gamma}_0, \boldsymbol{\beta}_0 \in \mathbb{R}^d$ across nodes, and $\boldsymbol{\lambda}^{(o)}$ is $\boldsymbol{\lambda}$ rotated to start at λ_o. MLP_o is shared across objectives in MOTSP and objective-specific in MOCVRP, whose objectives share the same node features. Details are provided in Appendix B.

With $H^{(o,\ell)} \in \mathbb{R}^{N \times d}$ stacking the embeddings of objective o at layer ℓ, each of the L layers applies a multi-head attention (MHA) sublayer (Vaswani et al., 2017), whose queries and keys are rotated by the positional information $\boldsymbol{p}^{(o)}$ (Section 5.2), and a feed-forward (FF) sublayer, both with a residual connection (He et al., 2016) and layer normalisation (LN) (Ma et al., 2021):

$$\begin{aligned} \widehat{H}^{(o,\ell)} &= \mathrm{LN}\left(H^{(o,\ell-1)} + \mathrm{MHA}_\ell\left(H^{(o,\ell-1)}; \boldsymbol{p}^{(o)}\right)\right), \\ H^{(o,\ell)} &= \mathrm{LN}\left(\widehat{H}^{(o,\ell)} + \mathrm{FF}_\ell\left(\widehat{H}^{(o,\ell)}\right)\right), \end{aligned} \tag{7}$$

for $\ell = 1, \dots, L$, with $d = 128$, $L = 3$, four heads of dimension $d_h = 32$, and hidden ReLU layers of dimension 128 in the input MLP and 64 in the FF sublayer. The preference is introduced into the encoder only through Eq. (6), and all layers are shared across objectives and preferences.

5.2 SIZE-ALIGNED CYCLIC ROTARY POSITIONAL ENCODING

Circularity (Ma et al., 2021) and, when a solution has multiple routes, *hierarchy* (Hua et al., 2026) are two desirable properties of a positional encoding for routing solutions. For a single model to serve all problem sizes, the encoding needs also to be *size-aligned*, i.e., transfer across sizes, including larger ones unseen in training. Building on rotary positional encoding (Su et al., 2024), we design a size-aligned cyclic rotary positional encoding (SC-RoPE) that satisfies all three.

For *circularity*, we replace $\boldsymbol{R}(p)$ in Eq. (4) with $\boldsymbol{R}_N(p)$, whose u-th rotation pair uses the angle:

$$\varphi_u(p) = \frac{2\pi\kappa_u(N)}{N} p, \tag{8}$$

where $u = 1, \dots, U$ with $U = d_h/2$ and $\kappa_u(N)$ is a positive integer, so that $\boldsymbol{R}_N(p+N) = \boldsymbol{R}_N(p)$ while attention still depends only on position differences.

For *hierarchy*, when a solution consists of several routes, let $r(i)$ denote the route containing node i, s_i the cumulative travel distance from its departure depot to i, and $D_{r(i)}$ the length of that route:

$$\tau_i = \frac{s_i}{\max\{D_{r(i)}, \epsilon\}}, \qquad \varphi_u^{\mathrm{route}}(\tau_i) = 2\pi u \tau_i. \tag{9}$$

The total-distance stream uses Eq. (8) and the makespan stream Eq. (9), so that both a node's position in the complete solution and its position within its route are represented.

For *size alignment*, the rotation pairs form two groups. The first G keep a fixed number of turns over the solution, so their wavelengths stay a fixed fraction of it, while the remaining ones scale their harmonics with N so that their wavelengths span a fixed number of nodes:

$$\kappa_u(N) = \begin{cases} u, & 1 \le u \le G, \\ \mathrm{clip}_{[1, \lfloor N/2 \rfloor]}\left(\left\lfloor \bar{\kappa}_u \frac{N}{N_{\mathrm{ref}}} + \frac{1}{2} \right\rfloor\right), & G < u \le U, \end{cases} \tag{10}$$

where $G = 4$, N_{ref} is a reference length and $\bar{\kappa}_u = 2(U/2)^{(u-G-1)/(U-G-1)}$ is spaced logarithmically from 2 to U. Up to rounding and clipping, $N/\kappa_u(N) \approx N_{\mathrm{ref}}/\bar{\kappa}_u$, so these wavelengths are comparable across sizes. All harmonics are computed from N and add no parameters. See Appendix C for technical details.

5.3 Score-Level Fusion

The objective-wise representation (Section 5.1) offers a further entry point for the preference: where the scores of a move under different objectives are fused. SUPL2I fuses the streams only there, at the level of pairwise compatibility scores, and conditions the fusion on the preference. For stream o and head η, a multi-head compatibility (MHC) sublayer (Ma et al., 2021) produces

$$c_{ij}^{(o,\eta)} = \frac{(A_\eta \boldsymbol{h}_i^{(o,L)})^\top (B_\eta \boldsymbol{h}_j^{(o,L)})}{\sqrt{d_h}}, \tag{11}$$

where $A_\eta, B_\eta \in \mathbb{R}^{d_h \times d}$ are shared across streams, with four heads as in the encoder. Eq. (11) scores the node pair (i, j) under each objective's own embeddings; concatenating these scores over streams and heads gives $\boldsymbol{c}_{ij} \in \mathbb{R}^{4m}$. Tri-TSP additionally uses gated cross-objective scores (see Appendix B).

An MLP with two hidden layers of 32 dimensions maps $\boldsymbol{c}_{ij}$ to move scores, each hidden layer modulated by the preference as in Eq. (6), with scale and shift vectors:

$$(\boldsymbol{\gamma}_1, \boldsymbol{\beta}_1, \boldsymbol{\gamma}_2, \boldsymbol{\beta}_2) = W_{\text{cond}} \boldsymbol{\lambda} + \boldsymbol{b}_{\text{cond}}, \tag{12}$$

where $W_{\text{cond}} \in \mathbb{R}^{128 \times m}$ and $\boldsymbol{b}_{\text{cond}} \in \mathbb{R}^{128}$. With $\boldsymbol{z}_{ij}^{(0)} = \boldsymbol{c}_{ij}$, the MLP produces:

$$\boldsymbol{z}_{ij}^{(r)} = \text{ReLU}\Big((\mathbf{1} + \boldsymbol{\gamma}_r) \odot F_r(\boldsymbol{z}_{ij}^{(r-1)}) + \boldsymbol{\beta}_r\Big), \qquad \boldsymbol{u}_{ij} = F_3(\boldsymbol{z}_{ij}^{(2)}) \in \mathbb{R}^{|\Omega|}, \tag{13}$$

for $r = 1, 2$, where F_1, F_2, F_3 are affine layers shared across preferences, $\boldsymbol{\gamma}_r, \boldsymbol{\beta}_r \in \mathbb{R}^{32}$ modulate hidden layer r and are shared across node pairs, and Ω is the operator set. For action $a = (\omega, i, j)$ with $\omega \in \Omega$, we bound the logit as $\ell_a = C \tanh(u_{ij,\omega})$ with $C = 6$, following Ma et al. (2021). After masking, a joint softmax over node pairs and operators defines $\pi_\theta(a \mid s_t, \boldsymbol{\lambda})$.

5.4 Training and Inference

To train the model, we adopt n-step proximal policy optimisation (PPO) (Schulman et al., 2017) with a shared baseline (Ye et al., 2025). Specifically, we train π_θ on instances of different sizes using M trajectories per instance. Each trajectory $j \in \{1, \ldots, M\}$ uses a preference $\boldsymbol{\lambda}_j$ sampled uniformly from Δ^{m-1} and held fixed throughout the trajectory. Transitions follow Eq. (5) to allow local moves between incomparable solutions, while rewards evaluate improvements in each trajectory's best scalarised objective value. Details are provided in Appendix D.1.

During inference, given a preference set $\Lambda = \{\boldsymbol{\lambda}_j\}_{j=1}^W$, we run W parallel trajectories for T steps from random solutions. Trajectory j samples local moves from the trained policy π_θ under its fixed preference $\boldsymbol{\lambda}_j$, with a sampling temperature annealed linearly from 1.5 to 1.0; a sampled move is accepted unless the current solution dominates the resulting solution. The trajectory keeps an elite $\mathbf{x}_j^*$, i.e., its best solution so far under $g_{\boldsymbol{\lambda}_j}$. To share progress across preferences, every S steps each trajectory compares its elite with those of the other trajectories under $g_{\boldsymbol{\lambda}_j}$ and, if another is strictly better, adopts it as both its current solution and its elite. The non-dominated set of all solutions visited by the W trajectories is reported (Appendix D.2).

6 Experiments

6.1 Experimental Settings

Problems. We evaluate the proposed method on two commonly used MOCOPs: the multi-objective travelling salesman problem (MOTSP) (Lust & Teghem, 2010) and the multi-objective capacitated vehicle routing problem (MOCVRP) (Zajac & Huber, 2021) (see Appendix A). Following the common settings (Ma et al., 2021; Lin et al., 2022), three common sizes are considered, i.e., $n = 20/50/100$.

Baselines. Our methods are compared with four classes of strong baselines. Where applicable, these methods adopt the weighted sum (WS) scalarisation with 101 and 105 for bi-objective and tri-objective problems (Das & Dennis, 1998), respectively, for fair comparisons. (1) *L2C*: The single-model neural methods include **PMOCO** (Lin et al., 2022), **MORAM** (Wang et al., 2023), **NHDE** (Chen et al., 2023b), and the state-of-the-art **CNH** (Fan et al., 2025) and **WE-CA** (Chen et al., 2025). Both CNH

and WE-CA are unified models trained across problem size $n \in \{20, 21, \ldots, 100\}$. The multi-model neural methods include **DRL-MOA** (Li et al., 2020), and **EMNH** (Chen et al., 2023a). (2) *L2I*: **MOGLS+WRPN** (Ye et al., 2025) with a population size of 100 and $T \in \{2000, 5000\}$ iterations, following the original settings. (3) *Conventional methods*: The MOEAs include **MOEA/D** (Zhang & Li, 2007) and **NSGA-II** (Deb et al., 2002), both implemented on Pymoo[1] with a search budget of 5×10^5 function evaluations per instance. Local search or hybrid methods are **MOGLS** (Jaszkiewicz, 2002) with 4,000 iterations and **PPLS/D-C** (Shi et al., 2022) with 200 iterations, using 2-opt operator for MOTSP (Chen et al., 2025) and 2-opt, relocate, and exchange for MOCVRP (Ye et al., 2025). (4) *Strong classical solvers*: They include **WS-LKH** and **WS-ORT**, which solve each weighted-sum subproblem with LKH (Helsgaun, 2000; Tinós et al., 2018) and OR-Tools[2], respectively.

Training. We train one model per problem for 200 epochs of 10 batches, each batch sampling a size $n \in \{20, 21, \ldots, 100\}$ uniformly with a size-dependent number of instances. We use the 5-step PPO (Ma et al., 2021) with $T_{\text{train}} = 200$, clip range $\epsilon = 0.1$, entropy weight 10^{-3}, gradient-norm clipping at 0.04, and discount $\gamma = 0.999$, and Adam (Kingma & Ba, 2015) with learning rate 10^{-4}, decayed by 0.98 per epoch after epoch 40 (see Appendix D.1).

Inference and metrics. At inference, each instance is solved by parallel preference trajectories for $T \in \{1000, 5000\}$ steps (Ma et al., 2021), with the sampling temperature (Bello et al., 2017) annealed from 1.5 to 1.0 and elite sharing every 200 steps. We evaluate all methods using three metrics: average hypervolume (HV) (Zitzler & Thiele, 1999), average HV gap with respect to SUPL2I-Aug (T=5k), and total runtime on four GPUs. HV is used as the primary quality indicator, where higher HV means better comprehensive performance in terms of convergence and diversity (Li & Yao, 2019). Methods with the "-Aug" suffix use instance augmentation (Lin et al., 2022). For MOTSP, L2C baselines use 64 augmentations as in their original papers (Lin et al., 2022; Chen et al., 2025); L2I methods use 8 as in (Ma et al., 2021; Ye et al., 2025). A Wilcoxon signed-rank test (Wilcoxon, 1945) at the 1% significance level is used. The best results (including statistical equivalence) are shown in bold, while the second-best result are underlined. All experiments are implemented in Python on a machine with four NVIDIA RTX 5090 GPUs and an Intel Xeon Platinum 8570 CPU.

6.2 Experimental results

Comparison analysis. Tables 1 and 2 report the results on bi-objective problems (Bi-TSP and Bi-CVRP) and tri-objective ones (Tri-TSP), respectively. Overall, SUPL2I-Aug (T=5k) achieves the best HV results on all problems and sizes. Even with the small budget (T=1k), SUPL2I already outperforms all L2C methods on all MOTSP instances, including their "-Aug" versions at a comparable running time (e.g., 3.5 vs. 2.1 minutes for WE-CA-Aug on Bi-TSP100). Notably, with the larger budget (T=5k), SUPL2I surpasses the strong classical solver WS-LKH on all MOTSP instances in less time, whereas most neural methods fall behind WS-LKH, e.g., all 8 on Bi-TSP100. On Tri-TSP50, SUPL2I (T=5k) attains a much smaller gap of 0.54%, against 4.67% for WE-CA and 1.46% for WRPN (T=5k). For Bi-CVRP, SUPL2I (T=1k) surpasses WS-ORT at every size in less time, and outperforms WRPN at every size even when the latter is given more budget (T=5k). SUPL2I leads on Bi-CVRP20 with T=1k and on Bi-CVRP50 with T=5k; on Bi-CVRP100, SUPL2I-Aug (T=5k) attains the best HV, at a higher time cost than L2C methods.

Generalisation across problem sizes. We evaluate the generalisation ability of SUPL2I on the unseen larger sizes, i.e., Bi-TSP150 and Bi-TSP200. The results are reported in Table 5. All neural methods are trained on Bi-TSP100, except CNH, WE-CA and SUPL2I, which are trained across varying sizes $n \in \{20, 21, \ldots, 100\}$. As shown, SUPL2I outperforms all neural methods, while WRPN degrades sharply, which even augmentation does not remedy. Notably, SUPL2I (T=5k) surpasses WE-CA-Aug on Bi-TSP150. As shown in Table 6, SUPL2I (T=5k) outperforms WS-LKH on KroAB100 and, with augmentation, it even performs better on KroAB150.

Effectiveness of objective-wise representation. We now consider ablation studies of the proposed method. We first compare SUPL2I with a variant (w/o objective-wise repr.) that concatenates the node features of all objectives into a single representation, as in WRPN (Ye et al., 2025). Fig. 2a plots

[1] https://pymoo/org

[2] https://developers.google.com/optimization

Table 1: Performance on Bi-TSP and Bi-CVRP instances.

Method	Bi-TSP20			Bi-TSP50			Bi-TSP100		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
WS-LKH	0.6270	0.44%	10m	0.6415	0.58%	1.2h	0.7090	0.35%	7.0h
WS-ORT	0.6270	0.44%	8.2m	0.6381	1.11%	37m	0.7025	1.26%	3.1h
MOEA/D	0.6283	0.24%	13h	0.6325	1.97%	14h	0.6844	3.80%	15h
NSGA-II	0.6292	0.09%	3.7h	0.6189	4.08%	4.1h	0.5779	18.8%	4.9h
MOGLS	0.6294	0.06%	11m	0.6412	0.62%	8.2h	0.6980	1.89%	17h
PPLS/D-C	0.6288	0.16%	15m	0.6383	1.07%	3.3h	0.6837	3.90%	22h
DRL-MOA	0.6007	4.61%	0.5s	0.5762	10.7%	0.8s	0.6219	12.6%	1.3s
EMNH	0.6271	0.42%	0.2s	0.6373	1.23%	0.4s	0.6977	1.93%	1.3s
MORAM	0.6219	1.26%	0.1s	0.6255	3.06%	0.1s	0.6821	4.13%	0.2s
NHDE	0.6267	0.48%	0.5s	0.6370	1.28%	0.9s	0.6991	1.74%	2.6s
PMOCO	0.6263	0.55%	0.1s	0.6358	1.47%	0.3s	0.6986	1.81%	1.6s
CNH	0.6268	0.48%	0.2s	0.6368	1.31%	0.4s	0.6995	1.68%	1.6s
WE-CA	0.6270	0.44%	0.2s	0.6392	0.93%	0.4s	0.7034	1.13%	1.6s
EMNH-Aug	0.6277	0.33%	3.0s	0.6410	0.66%	20s	0.7026	1.24%	2.0m
NHDE-Aug	0.6271	0.43%	11s	0.6405	0.74%	35s	0.7036	1.10%	2.8m
PMOCO-Aug	0.6275	0.36%	4.8s	0.6403	0.77%	21s	0.7035	1.12%	2.0m
CNH-Aug	0.6275	0.36%	5.3s	0.6401	0.79%	23s	0.7035	1.12%	2.0m
WE-CA-Aug	0.6271	0.42%	5.3s	0.6413	0.60%	23s	0.7066	0.69%	2.1m
MOGLS+WRPN (T=2k)	0.6296	0.03%	2.1m	0.6433	0.29%	8.3m	0.7050	0.91%	9.0m
MOGLS+WRPN (T=5k)	0.6297	0.02%	5.4m	0.6438	0.22%	21m	0.7056	0.83%	22m
SUPL2I (T=1k)	**0.6298**	0.00%	22s	0.6447	0.08%	1.2m	0.7080	0.46%	3.5m
SUPL2I (T=5k)	**0.6298**	0.00%	1.8m	0.6451	0.02%	5.9m	0.7101	0.19%	25m
MOGLS+WRPN-Aug (T=2k)	**0.6298**	0.01%	8.9m	0.6447	0.08%	13m	0.7080	0.49%	35m
MOGLS+WRPN-Aug (T=5k)	**0.6298**	0.00%	22m	0.6449	0.05%	31m	0.7083	0.44%	1.6h
SUPL2I-Aug (T=1k)	**0.6298**	0.00%	2.8m	**0.6452**	0.00%	19m	0.7105	0.13%	36m
SUPL2I-Aug (T=5k)	**0.6298**	0.00%	36m	**0.6452**	0.00%	55m	**0.7115**	0.00%	2.8h

Method	Bi-CVRP20			Bi-CVRP50			Bi-CVRP100		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
WS-ORT	0.4287	0.52%	42m	0.4061	1.31%	3.7h	0.4022	1.50%	14h
MOEA/D	0.4251	1.35%	16h	0.4027	2.13%	21h	0.3967	2.85%	33h
NSGA-II	0.4268	0.96%	6.0h	0.4035	1.94%	11h	0.3721	8.87%	24h
MOGLS	0.4293	0.38%	1.7h	0.4043	1.74%	23h	0.4007	1.86%	92h
PPLS/D-C	0.4272	0.87%	33m	0.4037	1.89%	5.8h	0.3979	2.55%	43h
DRL-MOA	0.4204	2.44%	1.1s	0.4076	0.95%	2.2s	0.4053	0.73%	4.1s
EMNH	0.4297	0.28%	0.3s	0.4096	0.46%	0.6s	0.4071	0.30%	2.1s
NHDE	0.4282	0.64%	0.7s	0.4048	1.61%	1.3s	0.3931	3.73%	3.5s
PMOCO	0.4276	0.76%	0.3s	0.4053	1.51%	0.5s	0.3691	9.60%	1.9s
CNH	0.4288	0.49%	0.4s	0.4086	0.71%	0.6s	0.4067	0.38%	2.0s
WE-CA	0.4290	0.44%	0.4s	0.4090	0.61%	0.7s	0.4069	0.35%	2.1s
EMNH-Aug	0.4301	0.18%	0.5s	0.4105	0.25%	4.0s	0.4080	0.07%	18s
NHDE-Aug	0.4299	0.24%	1.7s	0.4087	0.67%	5.0s	0.3988	2.34%	24s
PMOCO-Aug	0.4288	0.49%	0.7s	0.4081	0.83%	3.0s	0.3807	6.75%	17s
CNH-Aug	0.4299	0.25%	0.8s	0.4100	0.37%	3.3s	0.4077	0.15%	18s
WE-CA-Aug	0.4301	0.20%	0.8s	0.4104	0.27%	3.3s	0.4081	0.04%	18s
MOGLS+WRPN (T=2k)	0.4307	0.06%	3.2m	0.4097	0.44%	14m	0.4034	1.20%	54m
MOGLS+WRPN (T=5k)	0.4308	0.03%	11m	0.4101	0.34%	32m	0.4044	0.96%	1.8h
SUPL2I (T=1k)	0.4308	0.03%	35s	0.4101	0.32%	2.1m	0.4047	0.89%	5.5m
SUPL2I (T=5k)	**0.4309**	0.01%	3.0m	0.4110	0.11%	24m	0.4069	0.36%	48m
MOGLS+WRPN-Aug (T=2k)	**0.4309**	0.01%	15m	0.4107	0.19%	35m	0.4052	0.76%	1.2h
MOGLS+WRPN-Aug (T=5k)	**0.4309**	0.01%	37m	0.4110	0.13%	1.3h	0.4060	0.56%	2.8h
SUPL2I-Aug (T=1k)	**0.4309**	0.01%	4.5m	0.4112	0.07%	34m	0.4067	0.41%	47m
SUPL2I-Aug (T=5k)	**0.4309**	0.00%	23m	**0.4115**	0.00%	1.5h	**0.4083**	0.00%	3.6h

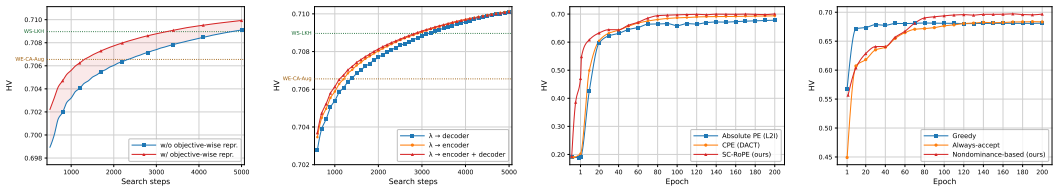
the HV trajectories on Bi-TSP100 over 5,000 search steps. The objective-wise representation (w/ objective-wise repr.) stays ahead at every budget, e.g., 0.7022 vs. 0.6989 at 500 steps and 0.7099 vs. 0.7091 at 5,000 steps. These results show that the objective-wise representation improves both search efficiency and the resulting solution quality.

Where to inject the preference. We compare three variants of SUPL2I on Bi-TSP100 that differ only in where the preference $\boldsymbol{\lambda}$ is injected: into the score fusion (decoder), into the node embeddings (encoder), or into both (encoder + decoder), respectively. As shown in Fig. 2b, all three methods converge to nearly the same HV with a large budget (T=5k), so the injection point decides not whether the search succeeds but how quickly it improves. Conditioning both is consistently best, and conditioning the encoder alone is better than conditioning the decoder alone. They overtake WS-LKH after about 3.2k (decoder), 2.9k (encoder) and 2.8k (encoder + decoder) steps, respectively.

Effectiveness of SC-RoPE. We compare SUPL2I (SC-RoPE) with two variants that differ only in the positional encoding: CPE of DACT (Ma et al., 2021), and Absolute PE of Wu et al. (2021)). As shown in Fig. 2c, on the Bi-TSP100 validation set, SUPL2I (SC-RoPE) improves from the first

Table 2: Performance on Tri-TSP instances.

Method	Tri-TSP20			Tri-TSP50			Tri-TSP100		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
WS-LKH	0.4712	1.44%	11m	0.4440	3.56%	1.2h	0.5076	2.03%	7.6h
WS-ORT	0.4706	1.57%	8.0m	0.4356	5.39%	38m	0.4958	4.30%	3.3h
MOEA/D	0.4757	0.50%	16h	0.4261	7.45%	21h	0.4311	16.8%	22h
NSGA-II	0.4696	1.78%	5.0h	0.3125	32.1%	6.3h	0.2184	57.9%	7.8h
MOGLS	0.4758	0.48%	20m	0.4200	8.77%	3.3h	0.4367	15.7%	12h
PPLS/D-C	0.4766	0.31%	1.5h	0.4312	6.34%	14h	0.4460	13.9%	81h
DRL-MOA	0.4495	5.98%	0.4s	0.3894	15.4%	0.8s	0.3254	37.2%	1.6s
EMNH	0.4698	1.74%	0.3s	0.4325	6.06%	0.4s	0.4866	6.08%	1.6s
MORAM	0.4576	4.29%	0.1s	0.4097	11.0%	0.1s	0.4585	11.5%	0.2s
PMOCO	0.4692	1.85%	0.3s	0.4324	6.07%	0.3s	0.4895	5.53%	1.6s
CNH	0.4677	2.18%	0.3s	0.4255	7.59%	0.5s	0.4811	7.14%	1.7s
WE-CA	0.4707	1.57%	0.3s	0.4389	4.67%	0.4s	0.4975	3.98%	1.7s
EMNH-Aug	0.4712	1.45%	4.3s	0.4400	4.43%	21s	0.4946	4.54%	2.1m
PMOCO-Aug	0.4712	1.45%	5.0s	0.4399	4.46%	23s	0.4957	4.32%	2.0m
CNH-Aug	0.4702	1.65%	5.5s	0.4324	6.08%	23s	0.4883	5.74%	2.1m
WE-CA-Aug	0.4712	1.44%	5.5s	0.4425	3.89%	24s	0.5021	3.08%	2.1m
MOGLS+WRPN (T=2k)	0.4759	0.46%	8.9m	0.4510	2.04%	9.1m	0.5071	2.12%	9.2m
MOGLS+WRPN (T=5k)	0.4766	0.31%	22m	0.4537	1.46%	23m	0.5101	1.54%	23m
SUPL2I (T=1k)	0.4779	0.04%	1.4m	0.4543	1.32%	6.4m	0.5050	2.53%	16m
SUPL2I (T=5k)	**0.4781**	0.00%	8.6m	0.4579	0.54%	24m	0.5120	1.18%	33m
MOGLS+WRPN-Aug (T=2k)	0.4774	0.15%	9.3m	0.4574	0.64%	12m	0.5154	0.52%	36m
MOGLS+WRPN-Aug (T=5k)	0.4776	0.10%	24m	0.4587	0.36%	31m	0.5170	0.21%	1.5h
SUPL2I-Aug (T=1k)	**0.4781**	0.00%	4.9m	0.4587	0.36%	34m	0.5134	0.91%	51m
SUPL2I-Aug (T=5k)	**0.4781**	0.00%	49m	**0.4604**	0.00%	1.4h	**0.5181**	0.00%	4.2h



(a) Representation (b) Preference injection (c) Positional encoding (d) Acceptance

Figure 2: Ablation studies: (a) and (b) validate the effectiveness of the objective-wise representation and preference injection; (c) and (d) verify the effects of positional encodings and acceptance rules.

epoch, whereas both variants stay near the HV of the random initial solutions for about five epochs before rising; SC-RoPE stays ahead throughout and ends highest (0.701 vs. 0.694 for CPE and 0.678 for Absolute PE). Since SC-RoPE rotates the queries and keys directly, positions shape attention from the first update without any parameters to learn, which may explain the fast start. These results indicate that SC-RoPE both accelerates training and improves final performance.

Nondominance-based acceptance. We compare three acceptance rules: the nondominance-based; always-accept (Wu et al., 2021); and greedy that accepts a candidate only if it improves the scalarised objective. Fig. 2d plots the validation HV curves on Bi-TSP100 during training. The greedy strategy rises fastest but soon plateaus, becoming stuck in local optima. Always-accept starts far lower, improves slowly, and ends at about the level of greedy: since every new solution is accepted, the search may drift into dominated solutions and need more steps to recover from them. The nondominance-based acceptance surpasses both after about 60 epochs and continues improving. These results show that the dominance-based rule achieves better performance than its alternatives.

7 Conclusion

This paper proposes SUPL2I, a standalone L2I method for MOCOPs in which a single model serves all preferences and problem sizes without the need for an external search framework. Extensive results show that SUPL2I outperforms state-of-the-art neural methods in both in-distribution and cross-size generalisation settings. It even outperforms the strong classical solver WS-LKH on MOTSP in the in-distribution setting. Notably, the proposed positional encoding can also be applied to single-objective L2I and to other tasks with cyclic or hierarchical structure. Lastly, it is worth mentioning that like other L2I methods, the proposed SUPL2I needs more inference time than L2C methods. Another limitation is that its memory grows linearly with the number of objectives as it keeps one embedding stream per objective.

Reproducibility Statement

Our code and datasets are available at https://anonymous.4open.science/r/SUPL2I-C518.

Moreover, we report the details needed to reproduce our implementation in Appendix A–E.

AI Use Statement

In this work, we used generative AI tools only to polish the writing to enhance readability. We have not used generative AI tools to generate synthetic data, to develop the theoretical or conceptual framework, to formulate or prove mathematical claims, to propose or refine hypotheses, to design or provide feedback on research methodology or experiments, to implement methods, or to interpret results; translation, dataset cleaning, and qualitative data analysis are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

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.

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, 2021.

Federico Berto, Chuanbo Hua, Junyoung Park, Laurin Luttmann, Yining Ma, Fanchen Bu, Jiarui Wang, Haoran Ye, Minsu Kim, Sanghyeok Choi, et al. RL4CO: an extensive reinforcement learning for combinatorial optimization benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

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*, 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*, 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*, 2025.

Tao Chen and Miqing Li. The weights can be harmful: Pareto search versus weighted search in multi-objective search-based software engineering. *ACM Transactions on Software Engineering and Methodology*, 32(1):1–40, 2023.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2019.

Indraneel Das and John E Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8 (3):631–657, 1998.

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

Matthias Ehrgott. *Multicriteria optimization*. Springer, 2005.

Matthias Ehrgott and Xavier Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *TOP*, 12(1):1–63, 2004.

Matthias Ehrgott, Xavier Gandibleux, and Anthony Przybylski. Exact methods for multi-objective combinatorial optimisation. In *Multiple criteria decision analysis: State of the art surveys*, pp. 817–850. Springer, 2016.

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, 2025.

Mingfeng Fan, Jianan Zhou, Yifeng Zhang, Yaoxin Wu, Jinbiao Chen, and Guillaume Sartoretti. Preference-driven multi-objective combinatorial optimization with conditional computation. In *Advances in Neural Information Processing Systems*, 2026.

Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Stephan Helfrich, Tyler Perini, Pascal Halffmann, Natashia Boland, and Stefan Ruzika. Analysis of the weighted tchebycheff weight set decomposition for multiobjective discrete optimization problems. *Journal of Global Optimization*, 86(2):417–440, 2023.

Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

André Hottung and Kevin Tierney. Neural large neighborhood search for routing problems. *Artificial Intelligence*, 313:103786, 2022.

André Hottung, Paula Wong-Chung, and Kevin Tierney. Neural deconstruction search for vehicle routing problems. *Transactions on Machine Learning Research*, 2025.

Chuanbo Hua, Federico Berto, André Hottung, Nayeli Gast Zepeda, Yining Ma, Zihan Ma, Paula Wong-Chung, Changhyun Kwon, Cathy Wu, Kevin Tierney, and Jinkyoo Park. Rethinking positional encoding for neural vehicle routing. *arXiv preprint arXiv:2605.11910*, 2026.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-Garcıa, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. In *International Conference on Machine Learning*, pp. 14631–14653. PMLR, 2023.

Andrzej Jaszkiewicz. Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71, 2002.

Zak Khan and Azam Asilian Bidgoli. A hierarchical importance-guided multi-objective evolutionary framework for deep neural network pruning. *Computers & Operations Research*, pp. 107602, 2026.

Minjun Kim, Junyoung Park, and Jinkyoo Park. Learning to cross exchange to solve min-max vehicle routing problems. In *International Conference on Learning Representations*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

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*, 2020.

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

Miqing Li and Xin Yao. Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–38, 2019.

Miqing Li, Xiaofeng Han, Xiaochen Chu, and Zimin Liang. Empirical comparison between moeas and local search on multi-objective combinatorial optimisation problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 547–556, 2024.

Sirui Li, Zhongxia Yan, and Cathy Wu. Learning to delegate for large-scale vehicle routing. In *Advances in Neural Information Processing Systems*, 2021.

Zimin Liang and Miqing Li. Random is faster than systematic in multi-objective local search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

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

Hao Lu, Xingwen Zhang, and Shuang Yang. A learning-based iterative method for solving vehicle routing problems. In *International Conference on Learning Representations*, 2019.

Fu Luo, Xi Lin, Mengyuan Zhong, Fei Liu, Zhenkun Wang, Jianyong Sun, and Qingfu Zhang. Learning to insert for constructive neural vehicle routing solver. In *Advances in Neural Information Processing Systems*, 2026.

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

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in Neural Information Processing Systems*, 2021.

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Hongliang Guo, Yuejiao Gong, and Yeow Meng Chee. Efficient neural neighborhood search for pickup and delivery problems. In *International Joint Conference on Artificial Intelligence*, pp. 4776–4784, 2022.

Yining Ma, Zhiguang Cao, and Yeow Meng Chee. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. In *Advances in Neural Information Processing Systems*, 2023.

Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.

Wenbin Ouyang, Sirui Li, Yining Ma, and Cathy Wu. Learning to segment for vehicle routing problems. In *International Conference on Learning Representations*, 2026.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA Journal on Computing*, 3(4): 376–384, 1991.

Peter Reiter and Walter J Gutjahr. Exact hybrid algorithms for solving a bi-objective vehicle routing problem. *Central European Journal of Operations Research*, 20(1):19–43, 2012.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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, 2021.

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, 2022.

Ralph E Steuer and Eng-Ung Choo. An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(3):326–344, 1983.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Rhea Sanjay Sukthanker, Arber Zela, Benedikt Staffler, Samuel Dooley, Josif Grabocka, and Frank Hutter. Multi-objective differentiable neural architecture search. In *International Conference on Learning Representations*, 2025.

Renato Tinós, Keld Helsgaun, and Darrell Whitley. Efficient recombination in the lin-kernighan-helsgaun traveling salesman heuristic. In *International Conference on Parallel Problem Solving from Nature*, pp. 95–107. Springer, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643, 2022.

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, 2023.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

Hong Wu, Jiahai Wang, and Zizhen Zhang. MODRL/D-AM: Multiobjective deep reinforcement learning algorithm using decomposition and attention model for multiobjective optimization. In *International Symposium on Intelligence Computation and Applications*, pp. 575–589. Springer, 2019.

Xuan Wu, Jinbiao Chen, Yang Li, Lijie Wen, Chunguo Wu, Yuanshu Li, Yubin Xiao, Chunyan Miao, You Zhou, and Di Wang. Wecon: An efficient weight-conditioned neural solver for multi-objective combinatorial optimization problems. *arXiv preprint arXiv:2605.22876*, 2026.

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, 2021.

Te Ye, Zizhen Zhang, Qingfu Zhang, Jinbiao Chen, and Jiahai Wang. Solving multiobjective combinatorial optimization via learning to improve method. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 9(3):2122–2136, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, 2026.

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.

Haotian Zhang, Jialong Shi, Jianyong Sun, and Zongben Xu. Learning to balance exploration and exploitation in pareto local search for multi-objective combinatorial optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 383–386, 2022a.

Haotian Zhang, Jialong Shi, Jianyong Sun, Qingfu Zhang, and Zongben Xu. Neuro-pls: A generalizable local search framework for multi-objective combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 2025.

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

Yongxin Zhang, Jiahai Wang, Zizhen Zhang, and Yalan Zhou. MODRL/D-EL: Multiobjective deep reinforcement learning with evolutionary learning for multiobjective optimization. In *International Joint Conference on Neural Networks*, pp. 1–8, 2021.

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, 2022b.

Guang Zhao, Byung-Jun Yoon, Gilchan Park, Shantenu Jha, Shinjae Yoo, and Xiaoning Qian. Pareto prompt optimization. In *International Conference on Learning Representations*, 2025.

Zhi Zheng, Shunyu Yao, Genghui Li, Linxi Han, and Zhenkun Wang. Pareto improver: Learning improvement heuristics for multi-objective route planning. *IEEE Transactions on Intelligent Transportation Systems*, 25(1):1033–1043, 2023.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

A Details of MOCOPs

A.1 MOTSP

Problem. An instance has n nodes, each with m pairs of coordinates $(x_i^{(o)}, y_i^{(o)}) \in [0,1]^2$, $o \in \{1,\dots,m\}$, one pair per objective, all drawn independently and uniformly (Lust & Teghem, 2010; Lin et al., 2022). A solution is a Hamiltonian cycle over the n nodes, and objective f_o is its Euclidean length measured in the o-th coordinates. Training instances are generated on the fly, and the test set contains 200 instances per size generated in the same way (Lin et al., 2022; Ye et al., 2025).

State. The solution $\mathbf{x}_t$ is the tour, $N = n$, and p_i is the position of node i along the tour, counted from an arbitrary fixed node. Stream o receives, for each node, the coordinates $(x_i^{(o)}, y_i^{(o)})$ and a recency channel $h_i \in [0,1]$, which is the same for all streams: at every accepted move, h_i is increased by 0.02 for all nodes (capped at 1) and reset to 0 for the two endpoints of the move; a rejected move leaves all h_i unchanged. The recency channel thus tells the policy which nodes have not been changed for a long time (Barrett et al., 2020). Initial solutions are uniformly random tours, and $h_i = 0$ initially.

Local operator and masking. $\Omega = \{\text{2-opt}\}$: the action (i, j) reverses the tour segment between nodes i and j. Masked actions are $i = j$, the previous move (i_{t-1}, j_{t-1}), and its reverse (j_{t-1}, i_{t-1}).

A.2 MOCVRP

Problem. Following the practice in neural combinatorial optimisation (Lin et al., 2022; Chen et al., 2025), an instance has a depot and n customers with coordinates drawn uniformly from $[0,1]^2$, and customer demands $d_i = \delta_i / D$ with δ_i drawn uniformly from $\{1,\dots,9\}$ and $D = 30$, 40, and 50 for $n = 20$, 50, and 100, which we extend to $n \in [20,40)$, $[40,70)$, and $[70,100]$ for the intermediate training sizes. The vehicle capacity is 1. A solution is a set of routes that start and end at the depot and together serve each customer exactly once, with the total demand of each route at most the capacity. The objectives are the total length of all routes f_1 and the makespan f_2, i.e., the length of the longest route (Zajac & Huber, 2021), both Euclidean. Training instances are generated on the fly, and the test set contains 200 instances per size generated in the same way (Lin et al., 2022).

Representation. Following Ma et al. (2021); Lin et al. (2022), a solution is a single cycle over $N = n + K$ nodes, the n customers and K copies of the depot, with $K = \lceil n(1+r) \rceil - n$ and $r = 0.5$, 0.4, and 0.2 for $n \in [20,40)$, $[40,70)$, and $[70,100]$, respectively ($K = 10$, 20, and 20 for $n = 20$, 50, and 100). Cutting the cycle at the depot copies gives at most K routes; two consecutive depot copies enclose an empty route of zero length and zero load, so any number of routes up to K can be represented. The position p_i is the index of node i along the cycle, counted from a fixed depot copy. Initial solutions are uniformly random permutations of the N nodes.

Penalised objectives and dominance. Capacity is handled as a soft constraint during the search. Let ℓ_r be the total demand of route r and $v = \sum_r \max\{0, \ell_r - 1\}$ the total overload. The penalised objectives are

$$f_1^p = f_1\Big(1 + \frac{v}{f_2}\Big), \qquad f_2^p = f_2 + v, \tag{14}$$

which coincide with (f_1, f_2) for feasible solutions. The transition of Eq. (5), the scalarisation $g_{\boldsymbol{\lambda}}$, the reward, and the elites are all computed on (f_1^p, f_2^p), so that infeasible solutions may be visited but are never rewarded over feasible ones of the same quality. Only feasible solutions enter the archive, at their true objectives (f_1, f_2).

Node features. Both streams receive the same 10 features per node, listed in Table 3, where $r(i)$ is the route containing node i; the makespan stream is distinguished by its own input MLP and by the route-local positional encoding of Eq. (9). The recency channel h_i is maintained as for MOTSP.

Local operators and masking. Follow the common settings in L2I (Wu et al., 2021; Ma et al., 2021; Ye et al., 2025), $\Omega = \{\text{2-opt}, \text{Relocate}, \text{Exchange}\}$, each acting on the cycle so that it remains a single cycle: 2-opt(i, j) reverses the segment between i and j, which covers both intra-route 2-opt

Table 3: Node features of MOCVRP.

Feature	Definition
Coordinates	(x_i, y_i)
Depot flag	1 for depot copies, 0 for customers
Demand	d_i (0 for depot copies)
Route load	$\ell_{r(i)}$
Route overload	$\max\{0, \ell_{r(i)} - 1\}$
Prefix load	cumulative demand along route $r(i)$ up to and including i (0 at depot copies)
Route length	$D_{r(i)}$
Relative route length	$D_{r(i)}/f_2$
Recency	h_i

and inter-route moves when the segment contains depot copies. Relocate(i, j) removes i and reinserts it after j, which also splits or merges routes when i or j is a depot copy. Exchange(i, j) swaps the positions of i and j, within or across routes. Statically masked actions are $i = j$ for every operator, and 2-opt and exchange with both i and j depot copies, which only relabel coinciding depot copies. Dynamically masked actions are relocate(i, j) with j the predecessor of i, which leaves the cycle unchanged, and, for the previous action, the same action and its inverse: the reverse pair for 2-opt and exchange, and relocating i_{t-1} back after its former predecessor for relocate.

B Details of the Proposed Model

Input MLPs. Each MLP_o in Eq. (6) is a two-layer perceptron with a hidden ReLU layer of $d = 128$ dimensions. Whether it is shared across objectives is decided by what separates the streams. In MOTSP, objective o is the tour length measured in its own coordinates; stream o therefore receives the coordinates of objective o. Since the inputs already differ across streams, a single MLP shared across objectives suffices and treats the objectives symmetrically. In MOCVRP, the total distance and the makespan are two functions of the same routes (the sum and the maximum of their lengths), so both streams receive the same node features. The streams are then distinguished by objective-specific MLPs and, for the makespan stream, by the route-local positional encoding of Eq. (9). Details of the input features of each problem are provided in Appendix A.2.

Preference injection in the encoder. Stream o receives $\boldsymbol{\lambda}^{(o)}$, obtained by cyclically shifting $\boldsymbol{\lambda}$ so that λ_o moves to the first position while the order of the other weights is preserved. For example, with $m = 3$ and $\boldsymbol{\lambda} = (0.5, 0.3, 0.2)$, the three streams receive $(0.5, 0.3, 0.2)$, $(0.3, 0.2, 0.5)$ and $(0.2, 0.5, 0.3)$, respectively. Because W_{in} is shared, its first column multiplies the first input entry of every stream; the shift guarantees that this entry is always the weight of the stream's own objective, so that one W_{in} applies the same rule to every stream. For $m = 2$, $\lambda_{o'} = 1 - \lambda_o$, so the modulation of stream o is an affine function of λ_o alone; for $m > 2$, the remaining positions additionally tell the stream how the rest of the weight is split among the other objectives.

Preference injection in the decoder. The compatibility scores in Eq. (11) depend on the preference through the encoder embeddings, while the MHC sublayer itself receives no additional preference input. In the decoder, the preference is injected directly into the fusion MLP through the modulation in Eq. (12) and, for Tri-TSP only, into the gate on the cross-objective scores described below. Both modulations, in the encoder and in the decoder, are initialised to zero, so that training starts from a preference-independent policy; the variants compared in Section 6.2 differ only in which modulation is trained and share the initialisation of all other parameters.

Normalisation. Following DACT (Ma et al., 2021), LN in Eq. (7) is not the standard per-node layer normalisation. For each instance and each stream, it subtracts the mean and divides by the standard deviation computed over all $N \times d$ entries of $H^{(o,\ell)}$, and it has no learnable scale or shift.

Score-level fusion. The MHC sublayer computes, for every node pair (i, j) and every head η, the score of Eq. (11) with $A_\eta, B_\eta \in \mathbb{R}^{32\times128}$; it has four heads and the same parameters for all streams. Each node pair is thus described by four scores per objective, which are concatenated into $\boldsymbol{c}_{ij} \in \mathbb{R}^{4m}$.

For Tri-TSP, the same MHC parameters additionally score every ordered pair of distinct objectives (o, o'), with the query taken from stream o and the key from stream o', which adds $4m(m-1)$ entries; these cross-objective entries are multiplied by a scalar gate $\tanh(\mathrm{MLP}_g(\boldsymbol{\lambda}))$, where $\mathrm{MLP}_g : \mathbb{R}^m \to \mathbb{R}^8 \to \mathbb{R}$ has a zero-initialised output layer, so that $\boldsymbol{c}_{ij} \in \mathbb{R}^{4m^2}$ in this case.

The fusion MLP turns this vector into one score per operator, $\mathbb{R}^{4m} \to \mathbb{R}^{32} \to \mathbb{R}^{32} \to \mathbb{R}^{|\Omega|}$, where $|\Omega| = 1$ for MOTSP (2-opt) and $|\Omega| = 3$ for MOCVRP (2-opt, relocate and exchange). Its two hidden layers are the ones modulated by the preference. Finally, each score is bounded by $C\tanh(\cdot)$ with $C = 6$ (Ma et al., 2021) before masking and the softmax over all (ω, i, j). Details of local operators are provided in Appendix A.2.

Stability analysis. Inference is stochastic (random start tours, sampled moves), so we repeat the default setting (no augmentation, T=1k, elite sharing every 200 steps) ten times with seeds 1–10 on the 200 randomly generated test instances of Bi-TSP100, Bi-CVRP100 and Tri-TSP. Figure 3 shows the ten per-run mean normalised HVs as box plots. The run-to-run range is 6×10^{-5} on Bi-TSP100, 2.8×10^{-4} on Bi-CVRP100, and 1.1×10^{-4} on Tri-TSP100, so a single run is representative in Section 6.2.

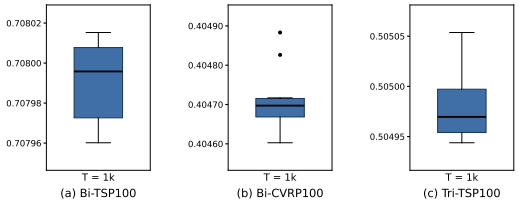


Figure 3: Normalised HV of 10 independent runs (seeds 1–10, no augmentation, T=1k) on (a) Bi-TSP100, (b) Bi-CVRP100 and (c) Tri-TSP100. Each box summarises the ten means of each run.

C Details of SC-RoPE

Implementation. As in RoPE (Su et al., 2024), only queries and keys are rotated; values are not. With $d_h = 32$, the query and key vectors of each head are split into two halves of 16 dimensions, and the u-th dimension of the first half is paired with the u-th dimension of the second half; each of the $U = 16$ pairs is rotated in its own plane by the u-th angle. The angles are recomputed at every step from the current solution and add no parameters.

Positions. For MOTSP, p_i is the index of node i in the tour and N the number of nodes; both streams use the same angles, since the tour is the same for all objectives. For MOCVRP, a solution is a giant tour in which depot copies separate the routes; p_i is the index in this sequence and N its length. In Eq. (9), the depot copy that starts a route has $s_i = 0$ and hence $\tau_i = 0$. The constant $\epsilon = 10^{-9}$ only matters for an empty route, i.e., a depot copy with no customers after it: its length is zero, and ϵ keeps the phase of this depot copy at $\tau_i = 0$. No size alignment is applied to Eq. (9): τ_i is already a fraction of its route, so every integer harmonic closes on the route.

Reference length. $N_{\mathrm{ref}} = 50$ for MOTSP and 80 for MOCVRP, whose giant tours contain the n customers plus K depot copies ($K = 10$ for Bi-CVRP20, and 20 for Bi-CVRP50 and Bi-CVRP100, following single-objective L2I (Ma et al., 2021)), so that the sequence length ranges from 30 to 120 during training; in both cases N_{ref} lies near the middle of the training range. For $N_{\mathrm{ref}} = 50$, the aligned wavelengths range from about 3 to 25 nodes. Without alignment ($\kappa_u = u$ for all u), the shortest wavelength would be $N/16$ nodes, i.e., about one node at $N = 20$ but about 12 nodes at $N = 200$, so the model would see different local resolutions at different sizes.

Algorithm 1 Nondominance-based PPO Training (one iteration)

Require: policy π_θ; curriculum depth T_{cl} (Eq. (15)); trajectory length T_{train}; update interval T_{u}; discount γ; reward scale c; clip range ϵ; entropy weight β

1: Sample a size $n \sim \mathcal{U}\{20, \dots, 100\}$ and a batch $\mathcal{B}$ of instances of size n
2: **for all** $I \in \mathcal{B}$ and $j \in \{1, \dots, W\}$ **do** ▷ in parallel; indices I, j omitted below
3: Sample $\boldsymbol{\lambda}_{I,j} \sim \mathcal{U}(\Delta^{m-1})$ and a random initial solution
4: Improve it with π_θ for T_{cl} steps under the transition of lines 9–10, without gradient
5: $\mathbf{x}_0 \leftarrow$ the improved solution; $\mathbf{x}_0^* \leftarrow \mathbf{x}_0$
6: **end for**
7: **for** $t = 0, \dots, T_{\mathrm{train}} - 1$ **do**
8: Sample $a_t = (\omega_t, i_t, j_t) \sim \pi_\theta(\cdot \mid s_t, \boldsymbol{\lambda})$ and store $\ell_t \leftarrow \log \pi_\theta(a_t \mid s_t, \boldsymbol{\lambda})$
9: $\tilde{\mathbf{x}}_{t+1} \leftarrow \mathcal{T}_{\omega_t}(\mathbf{x}_t, i_t, j_t)$ ▷ candidate solution
10: $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t$ **if** $\mathbf{x}_t \prec \tilde{\mathbf{x}}_{t+1}$, **else** $\tilde{\mathbf{x}}_{t+1}$ ▷ Eq. (5)
11: $r_t \leftarrow c \max\{0,\ g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x}_t^*)) - g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x}_{t+1}))\}$
12: $\mathbf{x}_{t+1}^* \leftarrow \arg\min_{\mathbf{x} \in \{\mathbf{x}_t^*, \mathbf{x}_{t+1}\}} g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x}))$
13: **if** $(t+1) \bmod T_{\mathrm{u}} = 0$ **then** ▷ PPO update on the last T_{u} steps
14: $R_\tau \leftarrow \sum_{k=\tau}^{t} \gamma^{k-\tau} r_k$ for $\tau = t - T_{\mathrm{u}} + 1, \dots, t$ ▷ bootstrapped by zero
15: $A_{I,j,\tau} \leftarrow R_{I,j,\tau} - \frac{1}{W} \sum_{j'=1}^{W} R_{I,j',\tau}$ ▷ shared baseline
16: $\hat{A} \leftarrow (A - \operatorname{mean} A)/(\operatorname{std} A + 10^{-8})$ ▷ over the window
17: $\rho_\tau \leftarrow \exp\big(\log \pi_\theta(a_\tau \mid s_\tau, \boldsymbol{\lambda}) - \ell_\tau\big)$
18: $\mathcal{L}(\theta) \leftarrow -\operatorname{mean}_\tau \min\{\rho_\tau \hat{A}_\tau,\ \operatorname{clip}(\rho_\tau, 1-\epsilon, 1+\epsilon)\hat{A}_\tau\} - \beta \operatorname{mean}_\tau \mathcal{H}[\pi_\theta(\cdot \mid s_\tau, \boldsymbol{\lambda})]$
19: Minimise $\mathcal{L}(\theta)$ over θ for three epochs
20: **end if**
21: **end for**

D Training and Inference

D.1 Training

Algorithm 1 gives one training iteration. Each iteration draws a size n uniformly from $\{20, 21, \dots, 100\}$ and a fresh batch of $B(n)$ instances of that size, where $B(n) = \operatorname{clip}\big(\operatorname{round}(C/n),\ B_{\min},\ B_{\max}\big)$ keeps the number of nodes per batch roughly equal to C, with $(C, B_{\min}, B_{\max}) = (1600, 16, 48)$ for Bi-TSP, $(1067, 10, 32)$ for Tri-TSP, and $(1920, 8, 32)$ for Bi-CVRP (16–48, 11–32, and 16–32 instances per batch).

For each instance, W preferences are sampled uniformly from Δ^{m-1} ($W = 101$ for the bi-objective problems and 105 for Tri-TSP (Das & Dennis, 1998)), each paired with a random initial solution. Following the curriculum learning of DACT (Ma et al., 2021), the initial solutions are first improved by the current policy for T_{cl} steps without gradient. T_{cl} grows from 0 to ξT_{train} with $\xi = 0.5$ along a sigmoid schedule over the $E = 2000$ training iterations (the 200 epochs of 10 batches in Section 6.1),

$$T_{\mathrm{cl}}(e) = \left\lfloor \xi\, T_{\mathrm{train}}\, \frac{\sigma\Big(10\big(\frac{e}{E-1} - \frac{1}{2}\big)\Big) - \sigma(-5)}{\sigma(5) - \sigma(-5)} \right\rfloor, \qquad e = 0, \dots, E-1, \tag{15}$$

where σ is the logistic function, so that later iterations start from solutions of a quality closer to that met at inference.

The trajectory then runs for $T_{\mathrm{train}} = 200$ steps with the transition of Eq. (5). The reward is that of Section 5.4, $r_t = c \max\{0,\ g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x}_t^*)) - g_{\boldsymbol{\lambda}}(\mathbf{f}(\mathbf{x}_{t+1}))\}$, scaled by $c = 100$, where $\mathbf{x}_0^* = \mathbf{x}_0$ is the warm-started initial solution. Every $T_{\mathrm{u}} = 5$ steps, the discounted returns of the last T_{u} steps are computed with $\gamma = 0.999$ (bootstrapped by zero, as no critic is used), and the shared baseline subtracts, for each trajectory, the mean return of the W trajectories of the same instance (Kwon et al., 2020; Ye et al., 2025).

The resulting advantages are standardised over the window and used in three PPO epochs with clip range $\epsilon = 0.1$ and entropy weight $\beta = 10^{-3}$; gradients are clipped to norm 0.04. For MOTSP, advantages are standardised, and the policy-gradient loss averaged, over live groups only, i.e., instances for which at least one of the W trajectories received a non-zero reward in the window, following the

Algorithm 2 Inference (one instance)

Require: trained policy π_θ; preferences $\Lambda = \{\boldsymbol{\lambda}_j\}_{j=1}^{W}$; number of steps T; sharing interval S; temperatures $\{\mu_t\}_{t=0}^{T-1}$

1: For each $j \in \{1, \dots, W\}$: sample a random solution $\mathbf{x}_j$ and set $\mathbf{x}_j^* \leftarrow \mathbf{x}_j$; $\mathcal{A} \leftarrow \{\mathbf{f}(\mathbf{x}_j)\}_{j=1}^{W}$
2: **for** $t = 0, \dots, T-1$ **do**
3: **for all** $j \in \{1, \dots, W\}$ **do** ▷ in parallel
4: Sample a move $a \sim \pi_\theta(\cdot \mid s_j, \boldsymbol{\lambda}_j)$ with temperature μ_t and apply it to $\mathbf{x}_j$, giving the candidate $\tilde{\mathbf{x}}_j$
5: $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{f}(\tilde{\mathbf{x}}_j)\}$
6: **if not** $\mathbf{x}_j \prec \tilde{\mathbf{x}}_j$ **then** $\mathbf{x}_j \leftarrow \tilde{\mathbf{x}}_j$ ▷ Eq. (5)
7: **if** $g_{\boldsymbol{\lambda}_j}(\mathbf{f}(\mathbf{x}_j)) < g_{\boldsymbol{\lambda}_j}(\mathbf{f}(\mathbf{x}_j^*))$ **then** $\mathbf{x}_j^* \leftarrow \mathbf{x}_j$ ▷ elite
8: **end for**
9: **if** $(t+1) \bmod S = 0$ **and** $t+1 < T$ **then** ▷ elite sharing, from the elites before this step
10: **for all** $j \in \{1, \dots, W\}$ **do**
11: $k \leftarrow \arg\min_{k' \in \{1,\dots,W\}} g_{\boldsymbol{\lambda}_j}(\mathbf{f}(\mathbf{x}_{k'}^*))$
12: **if** $g_{\boldsymbol{\lambda}_j}(\mathbf{f}(\mathbf{x}_k^*)) < g_{\boldsymbol{\lambda}_j}(\mathbf{f}(\mathbf{x}_j^*))$ **then** $\mathbf{x}_j \leftarrow \mathbf{x}_k^*$, $\mathbf{x}_j^* \leftarrow \mathbf{x}_k^*$
13: **end for**
14: **end if**
15: **end for**
16: **return** the non-dominated subset of $\mathcal{A}$

motivation of dynamic sampling in DAPO (Yu et al., 2026). The Adam optimiser (Kingma & Ba, 2015) with learning rate 10^{-4} is used, decayed by a factor of 0.98 every 10 iterations from iteration 400. One epoch takes about 18 minutes for MOTSP and 22 minutes for MOCVRP on one NVIDIA RTX 5090 GPU. Every 100 (20 for Tri-TSP and Bi-CVRP) iterations, the current model is evaluated with $T = 500$ on 25 randomly generated validation instances per size, and the checkpoint with the highest mean archive HV over $n \in \{20, 50, 100\}$ is used for all reported results.

D.2 Inference

Algorithm 2 gives the inference procedure for one instance. Instances are processed in parallel on the four NVIDIA RTX 5090 GPUs. The preference set Λ is the same uniform grid on Δ^{m-1} as used with $W = 101$ for the bi-objective problems, and $W = 105$ for Tri-TSP (Das & Dennis, 1998).

Each trajectory starts from a random solution and runs for T steps; at step $t \in \{0, \dots, T-1\}$, moves are sampled with temperature $\mu_t = 1.5 - 0.5\, t/(T-1)$. Elite sharing (lines 11–12) takes place every $S = 200$ steps except at the last step, is computed from the elites as they stand before the sharing step, and adopts another elite only if it is strictly better under the trajectory's own scalarisation. The archive $\mathcal{A}$ collects the objective vectors of all solutions generated along the W trajectories, and its non-dominated subset is the reported solution set, from which the hypervolume in the tables is computed. For MOCVRP, only feasible solutions enter $\mathcal{A}$, evaluated on their true objectives (Appendix A.2).

Instance augmentation. For one set of coordinates, the eight symmetries of the unit square give eight equivalent instances with identical distances and hence identical objective values (Kwon et al., 2020). In MOTSP, each objective has its own coordinates, so L2C methods transform the sets independently and use all combinations, e.g., $8^2 = 64$ for Bi-TSP (Lin et al., 2022; Chen et al., 2025); MOCVRP has a single set of coordinates, so only eight copies exist. Following the L2I practice (Ma et al., 2021; Ye et al., 2025), SUPL2I-Aug uses eight copies for both problems, obtained by transforming the coordinates of the first objective for MOTSP and the node coordinates for MOCVRP. It runs Algorithm 2 on the eight copies at once; for MOTSP, elite sharing is pooled over the $8W$ trajectories of an instance, while for MOCVRP it stays within each copy, and the archives of the eight copies are merged before the non-dominated subset is taken.

Table 4: Ideal and reference points for the hypervolume.

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

E Evaluation

Test sets. For each problem and size, the test set contains 200 instances generated as described in Appendix A, and the same instances are used for all methods. The instances of Bi-TSP150 and Bi-TSP200 are generated in the same way for the cross-size evaluation. KroAB100/150/200 are commonly used benchmarks form TSPLIB (Reinelt, 1991) in neural multi-objective combinatorial optimisation (Lin et al., 2022; Fan et al., 2025; Chen et al., 2025; Ye et al., 2025).

Hypervolume indicator. Given a solution set P in the objective space and a reference point $\mathbf{z}^{\mathrm{ref}}$ dominated by every point of interest, the hypervolume (HV) of P is the Lebesgue measure μ of the region dominated by P and bounded by $\mathbf{z}^{\mathrm{ref}}$ (Zitzler & Thiele, 1999),

$$\mathrm{HV}_{\mathbf{z}^{\mathrm{ref}}}(P) = \mu\big(\{\mathbf{q} \in \mathbb{R}^m \mid \exists\, \mathbf{p} \in P :\ \mathbf{p} \preceq \mathbf{q} \preceq \mathbf{z}^{\mathrm{ref}}\}\big), \tag{16}$$

where $\mathbf{p} \preceq \mathbf{q}$ means $p_i \leq q_i$ for all i. A larger HV indicates a solution set that is closer to the Pareto front and better spread over it. To make the values comparable across problems and sizes, we report the normalised HV

$$\overline{\mathrm{HV}}(P) = \frac{\mathrm{HV}_{\mathbf{z}^{\mathrm{ref}}}(P)}{\prod_{i=1}^{m} \left(z_i^{\mathrm{ref}} - z_i^*\right)} \in [0, 1], \tag{17}$$

where $\mathbf{z}^*$ is the ideal point. All methods share the same $\mathbf{z}^*$ and $\mathbf{z}^{\mathrm{ref}}$ for a problem. Table 4 reports the ideal points and reference points, commonly used in NCO (Chen et al., 2025; Ye et al., 2025).

F Additional Experimental Results

F.1 Cross-Size Generalisation Results

As mentioned in Section 6.2, the detailed results of our method and other methods on larger unseen instances are provided in Table 5, which exhibits the superior generalisation capability of our method.

F.2 Results on Benchmark Instances

As in Section 6.2, Table 6 reports the results of our method and other methods on three benchmark instances (Reinelt, 1991) (Time reports the total running time in one NVIDIA RTX 5090 GPU).

Table 5: Performance on Bi-TSP150 and Bi-TSP200 instances.

Method	Bi-TSP150			Bi-TSP200		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
WS-LKH	**0.7100**	-0.06%	8.0h	**0.7459**	-0.58%	17h
WS-ORT	0.7015	1.14%	9.0h	0.7375	0.55%	20h
MOEA/D	0.6644	6.37%	16h	0.6820	8.04%	17h
NSGA-II	0.4687	34.0%	5.8h	0.4352	41.3%	6.6h
MOGLS	0.6949	2.07%	31h	0.7301	1.55%	82h
PPLS/D-C	0.6766	4.65%	41h	0.6902	6.93%	93h
EMNH	0.6936	2.26%	13s	0.7266	2.02%	1.1m
MORAM	0.6707	5.48%	6.2s	0.7020	5.34%	7.2s
NHDE	0.6950	2.06%	18s	0.7262	2.08%	35s
PMOCO	0.6943	2.16%	13s	0.7274	1.92%	28s
CNH	0.6953	2.02%	14s	0.7282	1.81%	28s
WE-CA	0.7008	1.24%	14s	0.7346	0.94%	29s
EMNH-Aug	0.6986	1.55%	15m	0.7310	1.44%	13m
NHDE-Aug	0.7000	1.35%	16m	0.7310	1.43%	41m
PMOCO-Aug	0.6992	1.46%	15m	0.7319	1.31%	30m
CNH-Aug	0.6997	1.40%	15m	0.7323	1.25%	33m
WE-CA-Aug	0.7044	0.73%	15m	0.7381	0.48%	33m
MOGLS+WRPN (T=2k)	0.6830	3.75%	12m	0.5385	27.4%	16m
MOGLS+WRPN (T=5k)	0.6836	3.66%	30m	0.5422	26.9%	40m
SUPL2I (T=1k)	0.7046	0.70%	16m	0.7327	1.20%	24m
SUPL2I (T=5k)	0.7083	0.18%	44m	0.7392	0.32%	1.2h
MOGLS+WRPN-Aug (T=2k)	0.6905	2.69%	1.2h	0.5639	24.0%	2.0h
MOGLS+WRPN-Aug (T=5k)	0.6908	2.65%	3.0h	0.5678	23.4%	5.0h
SUPL2I-Aug (T=1k)	0.7079	0.23%	1.1h	0.7361	0.74%	2.2h
SUPL2I-Aug (T=5k)	<u>0.7096</u>	0.00%	4.9h	<u>0.7416</u>	0.00%	8.5h

Table 6: Performance on the benchmark instances.

Method	KroAB100			KroAB150			KroAB200		
	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓	HV↑	Gap↓	Time↓
WS-LKH	0.7023	0.35%	2.3m	<u>0.7017</u>	0.08%	1.2m	**0.7430**	-0.42%	2.2m
WS-ORT	0.6967	1.14%	5.7s	0.6930	1.32%	16s	0.7344	0.74%	36s
MOEA/D	0.6731	4.49%	4.5m	0.6512	7.27%	4.7m	0.6787	8.27%	4.9m
NSGA-II	0.5612	20.4%	1.4m	0.4406	37.3%	1.7m	0.4034	45.5%	1.9m
MOGLS	0.6908	1.97%	5.7m	0.6855	2.40%	18m	0.7270	1.74%	51m
PPLS/D-C	0.6731	4.49%	6.3m	0.6640	5.45%	40m	0.6805	8.03%	1.5h
EMNH	0.6914	1.88%	1.9s	0.6838	2.63%	3.0s	0.7217	2.45%	3.8s
MORAM	0.6724	4.58%	0.1s	0.6588	6.19%	0.1s	0.6959	5.94%	0.1s
PMOCO	0.6901	2.07%	1.5s	0.6856	2.38%	2.2s	0.7242	2.12%	2.9s
CNH	0.6916	1.86%	1.9s	0.6860	2.32%	2.6s	0.7244	2.10%	3.2s
WE-CA	0.6948	1.40%	2.4s	0.6924	1.41%	3.5s	0.7317	1.10%	4.5s
EMNH-Aug	0.6958	1.26%	2.5s	0.6895	1.83%	6.0s	0.7269	1.76%	15s
PMOCO-Aug	0.6955	1.31%	2.2s	0.6907	1.65%	5.6s	0.7280	1.61%	14s
CNH-Aug	0.6962	1.20%	2.6s	0.6902	1.72%	5.9s	0.7286	1.52%	15s
WE-CA-Aug	0.6990	0.82%	2.8s	0.6957	0.94%	6.0s	0.7349	0.68%	15s
MOGLS+WRPN (T=2k)	0.6983	0.91%	8.4m	0.6713	4.42%	8.0m	0.5287	28.5%	7.8m
MOGLS+WRPN (T=5k)	0.6986	0.87%	21m	0.6727	4.22%	20m	0.5316	28.2%	20m
SUPL2I (T=1k)	0.7018	0.41%	19s	0.6961	0.88%	27s	0.7327	0.98%	36s
SUPL2I (T=5k)	0.7032	0.22%	1.5m	0.6997	0.37%	2.2m	0.7385	0.19%	2.8m
MOGLS+WRPN-Aug (T=2k)	0.7010	0.53%	7.9m	0.6816	2.95%	8.1m	0.5615	24.1%	8.6m
MOGLS+WRPN-Aug (T=5k)	0.7011	0.51%	20m	0.6819	2.90%	20m	0.5635	23.8%	21m
SUPL2I-Aug (T=1k)	<u>0.7040</u>	0.11%	38s	0.6994	0.41%	1.3m	0.7314	1.15%	2.1m
SUPL2I-Aug (T=5k)	**0.7047**	0.00%	3.2m	**0.7023**	0.00%	6.3m	<u>0.7399</u>	0.00%	10m