UECO: A Unified Encoder for Neural Combinatorial Optimization with Iterative Edge-Node Reasoning

Wenzheng Pan[1†], **Shuyi Yan**[1], **Nuoyan Chen**[1], **Yiyang Qu**[1], **Jiale Ma**[12], **Junchi Yan**[12*]
[1]Sch. of Computer Science & Sch. of Artificial Intelligence, Shanghai Jiao Tong University
[2]Shanghai Innovation Institute
{pwz1121,cozease_y,cny123222,q12345,heatingma,yanjunchi}@sjtu.edu.cn

Abstract

Despite the promise of Neural Combinatorial Optimization (NCO), mainstream pipelines exhibit a dominant empirical coupling among decision paradigms, training objectives, problem classes, and backbone architectures. Though not formally inseparable, different solving streams often pair distinct neural architectures, making backbone-induced variation a confounding factor in generic model development and fair cross-paradigm evaluation. To address this, we propose UECO, a simple yet effective Unified Encoder for Combinatorial Optimization with a structure-aware attention mixture scheme, where a lightweight module defined upon graph-based locality is interleaved with conventional global node-focused attention blocks to dynamically evolve edge representations and inject local topological information into attention scores. Instead of fusing static edge features in a single step, UECO thereby integrates local and global messages in a progressive and problem-agnostic manner. As a shared encoder orthogonal to task-specific decoders, UECO works adaptively with Local Construction (LC), Global Prediction (GP), and Adaptive Expansion (AE) paradigms, across node- or edge-oriented problems, symmetric or asymmetric tasks, and sparse or dense graph representations, for supervised, reinforcement, or generative learning. Extensive controlled experiments show that, via encoder-only substitution, UECO yields broad and near-consistent gains in solution and generalization quality over diverse backbones and task scales on TSP, ATSP, CVRP, ACVRP, MIS, MaxClique, and MaxCut, with supplementary KP and FFSP results further supporting its applicability beyond graph-based problems. Code is available at https://github.com/Thinklab-SJTU/UECO.

1 Introduction

Combinatorial Optimization Problems (COPs), often NP-hard by nature, arise broadly in logistics, network design, scheduling, and scientific discovery [1–5], where decision variables are discrete and the solution space scales combinatorially with problem size. Recently, Neural CO (NCO) [6] has emerged as a compelling alternative to exact and heuristic solvers, leveraging machine learning and deep models to amortize computation, exploit data distributions, and accelerate problem solving [7, 8]. Despite the progress, existing literature exhibits a noticeable empirical coupling among *decision paradigms*, *training objectives*, *problem classes* and *backbone architectures*[1]. Specifically, Local Construction (LC) methods [9–14] are often paired with Reinforcement Learning (RL) and node-centric Transformers for routing-like tasks; Global Prediction (GP) [15–17] methods frequently adopt Graph Convolutional Networks (GCNs) [18] and Supervised Learning (SL) for COPs with relatively simple constraints (e.g., TSP and MIS); and solvers following the Adaptive Expansion

*Correspondence to: Junchi Yan. †Project lead, who is also a visiting student at Shanghai Innovation Institute.
[1]In Appendix A, we detail these established *paradigms* (A.1), *architectures* (A.2), and *objectives* (A.3) within NCO.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

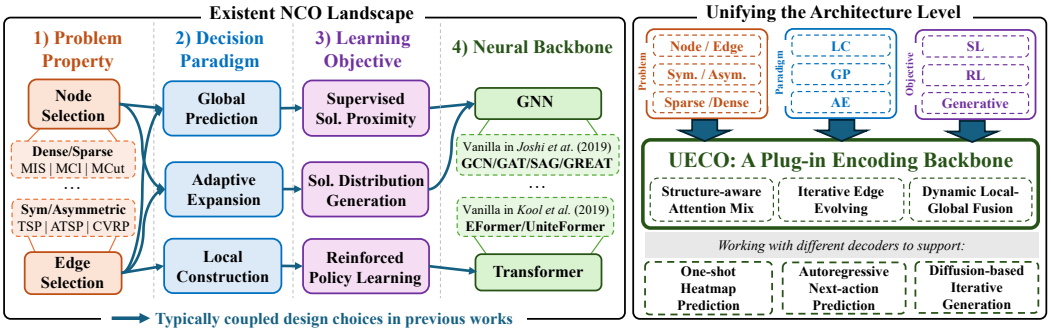


Figure 1: Position of UECO within the mainstream (graph-based) NCO landscape: a shared backbone connecting different solution streams in a comparable design and experimental space.

(AE) [19–22] paradigm consume global heatmaps predicted by generative models with dynamic decision granularity. These combinations reflect established engineering affinities rather than formal restrictions. However, as shown in Fig. 1, such prevailing combinations produce a fragmented design space in which architectural choices are repeatedly re-engineered across decision paradigms, problem properties, and learning objectives. Because moving between solution streams often changes the backbone together with these higher-level choices, backbone design becomes a confounding factor that complicates generic model development, controlled cross-paradigm ablation, and more independent examination of objectives and training mechanisms.

A principled remedy is to decouple representation learning from decision making by developing a *unified*, *high-performance*, *plug-and-play* encoder that serves as a shared foundation across paradigms, objectives, decoding strategies, and problem properties. Such an encoder must simultaneously i) preserve the global relational modeling capabilities of attention-based backbones, and ii) encode the crucial local topological (e.g. edge relations) inductive biases in graph-based COPs. Existing edge-aware attention models typically inject edge features into attention scores with static, one-shot additions [13, 23, 24], which limits their ability to progressively refine structural signals and adapt edge representations as node embeddings evolve across layers. In contrast, CO solvers that rely on GCNs [25], while effective at capturing neighborhood topology, face over-smoothing and struggle to translate increased parameter budgets into improved solution quality. Moreover, purely local aggregation underexploits global relational dependencies among distant nodes, which are naturally modeled by attention [26], further restricting their applicability to autoregressive RL settings.

Motivated by the above observations, we propose **UECO**, a Unified Encoder for Combinatorial Optimization, designed to minimize backbone-level discrepancy across NCO paradigms, and treat edge information as a first-class, evolving representation learned and updated throughout the attention stack. UECO introduces a new structure-aware attention mixture scheme: learnable local topology modules are interleaved between conventional node-focused attention blocks to dynamically update edge embeddings, which are then adaptively injected into the computation of the attention score at each layer. This iterative design progressively integrates local and global structure in a fully learnable and problem-agnostic manner, enabling the encoder to support diverse decoders (both simple greedy decoding for heatmap exploitation and learnable Transformer-style decoder for next-action prediction) and different training objectives (i.e., RL, SL, or generative learning). Inspired by [13, 24], UECO incorporates dual-channel attention modules that enable node and edge messages maintained and learnably propagate between the two endpoints of each edge, naturally accommodating asymmetric instances (e.g., ATSP/ACVRP) via directional representations. Major contributions of this paper are:

- **Shared Backbone for NCO.** We design UECO, a unified plug-and-play encoder that reduces dependence on paradigm-specific backbone choices and places LC, GP, and AE pipelines in a more consistent architectural space for controlled development and comparison.
- **Depth-wise Structure-aware Mixed-attention.** UECO interleaves local topology modules that persistently evolve edge representations with global node-centric attention blocks, using the evolved structure to modulate attention scores at every layer, rather than merging them via static additions.
- **Orthogonal yet Consistent Performance Gains.** Through encoder-level substitution, while retaining the host objectives, decoders, and feasibility mechanisms, UECO yields broad and near-consistent improvements across seven core graph problems and diverse learning settings, with supplementary KP and FFSP evidence extending the scope beyond graph-based tasks.

- **Cross-paradigm Insights.** Using this shared architectural substrate, we conduct controlled analyses of mechanism, scalability, and quality-efficiency trade-offs. Notably, the enhanced RL-based LC variant surpasses its generative counterparts on TSP-500 and transfers to even larger instances within divide-and-conquer pipelines, scaling up to TSP with 100K nodes and MIS with 11K nodes.

2 Preliminaries and Background

Notations. By convention, we use bold uppercase letters (e.g., $\mathbf{W}$) for matrices, bold lowercase (e.g., $\mathbf{x}$) for (column) vectors, denote by $[n]$ the set $\{1, \cdots, n\}$ for positive integer n, use superscript $\top$ to denote the transpose of a vector or a matrix, and adopt $||$ for tensor concatenation, $\sigma(\cdot)$ for the sigmoid function, and $\odot$ for element-wise multiplication.

Problem Interface. In this paper, we consider a broad class of COPs defined over graphs $\mathcal{G}$ with certain data-generating distribution $p(\mathcal{G})$, where each instance $\mathcal{I} \sim p(\mathcal{G})$ consists of a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$. Depending on the task, the solver may operate on i) *node features* $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_x}$ (e.g., 2D coordinates for TSP/CVRP), ii) *edge features* $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ (e.g., adjacency or pairwise weights, possibly asymmetric), or iii) both. Let N denote the number of decision variables for a certain task (usually $N = |\mathcal{V}|$ or $|\mathcal{E}|$) and $\Omega_{\mathcal{I}} \subseteq \{0,1\}^N$ denote the instance-specific feasible set, the objective is to find a discrete feasible (preferably optimal) solution $\mathbf{s}^* \in \Omega_{\mathcal{I}}$ that minimizes or maximizes a task-specific objective function $l(\mathbf{s}, \mathcal{I}) : \{0,1\}^N \to \mathbb{R}_{\geq 0}$. Full names and formal definitions of the COPs studied in this paper are provided in Appendix B.

3 Methodology

Overview. A *unified* encoder is supposed to produce expressive representations for the graph components, and can be paired *orthogonally* with different decision paradigms, training objectives, and task-specific decoders. In GP and AE paradigms, the unified encoder is supposed to predict a global probability heatmap over edges or nodes, e.g., $\mathbf{Z}_{\text{GP}} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times 2}$ (or $\mathbb{R}^{|\mathcal{V}| \times 2}$ for node-tasks), indicating the likelihood of each decision variable belonging to the solution set, which is then consumed by heuristic decoders. In the LC paradigm, the encoder generates informative contextual (node) embeddings which are further processed by learnable decoders [26, 11] to output the probability $\mathbf{z}_{\text{LC}} \in \mathbb{R}^{|\mathcal{V}|}$ over the next action to be taken.

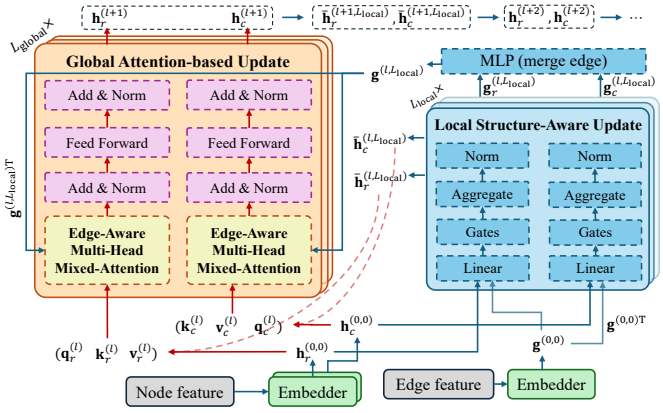


Figure 2: Overview of the UECO architecture.

In this light, UECO jointly maintains node and edge representations through a principled stacking of two complementary components: 1) a global, attention-based module as the primary backbone for updating node embeddings, and 2) a local, structure-aware module as an auxiliary system to maintain and refine edge (and optionally node) representations at each depth. The key design is to interleave these two components *layer by layer*, such that edge information is not injected in a static, one-shot manner, but is progressively evolved and incorporated into the global module via dynamic mixed attention scores as the representations deepen. Fig. 2 illustrates the overview of the architecture.

3.1 Modality-specific Input Embedders

UECO naturally accommodates different input modalities for COPs on graphs. Node features $\mathbf{X}$ are first mapped into the latent space through a learnable embedding function, $\mathbf{H}^0 = \phi(\mathbf{X}) \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $\phi(\cdot)$ is implemented as a linear projection optionally preceded by sinusoidal or one-hot-style positional encoding schemes [15, 13], and d is the hidden dimension throughout. Inspired by [21, 24], for problems with asymmetric structure or sensitive to directional signals (e.g., routing problems), UECO instantiates two separate yet parallel node streams, $\mathbf{H}_r^0$ and $\mathbf{H}_c^0$, corresponding to the *row* and *column* views from an edge weight matrix. On the other hand, edge features $\mathbf{E}$ are embedded

by a separate function $\mathbf{G}^0 = \psi(\mathbf{E}) \in \mathbb{R}^{|\mathcal{E}|\times d}$, where $\psi(\cdot)$ typically maps pairwise costs into a latent space using sinusoidal positional encodings followed by linear projection. By explicitly maintaining both embeddings from the input stage, UECO avoids collapsing structural information into a single type [11, 10] and enables subsequent cores to reason jointly over global context and local topology.

3.2 Global Node-centric Attention-based Updates

The backbone of UECO is a global attention module responsible for updating node representations. At layer $\ell \in [L_{\text{global}}]$, given node embeddings $\mathbf{H}^{\ell-1}$ (subscripts r, c are omitted without loss of generality) and edge embeddings $\mathbf{G}^{\ell-1}$, the queries, keys, and values are computed by

$$\mathbf{Q}_i = \mathbf{H}^{\ell-1}\mathbf{W}_i^Q,\ \mathbf{K}_i = \mathbf{H}^{\ell-1}\mathbf{W}_i^K,\ \mathbf{V}_i = \mathbf{H}^{\ell-1}\mathbf{W}_i^V, \tag{1}$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d\times d_h}$ are learnable parameter matrices, with i denoting the head index for multi-head attention (MHA) and d_h the dimension for each of the h heads (typically $d_h * h = d$). The mixed attention is updated following vanilla MHA, but its score is fused with evolved structural information $\bar{\mathbf{G}}^{\ell,L_{\text{local}}} \leftarrow \mathcal{F}_{\text{local}}(\cdot,\cdot,\mathbf{G}^{\ell-1})$, which is detailed in later sections. General formulation is:

$$\textbf{head}_i = \text{MixAtt}\left(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i; \bar{\mathbf{G}}^{\ell,L_{\text{local}}}\right),\ \text{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = (\textbf{head}_1 \,||\, \cdots \,||\, \textbf{head}_h)\mathbf{W}^O, \tag{2}$$

where $\mathbf{W}^O \in \mathbb{R}^{h\cdot d_h \times d}$ maps the concatenated heads back to the consistent dimension, which are then updated with residual connections, instance normalizations, and a feed-forward network:

$$\tilde{\mathbf{H}}^\ell \leftarrow \text{Norm}\big(\mathbf{H}^{\ell-1} + \text{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V})\big),\quad \mathbf{H}^\ell \leftarrow \text{Norm}\Big(\tilde{\mathbf{H}}^\ell + \text{FFN}(\tilde{\mathbf{H}}^\ell)\Big). \tag{3}$$

Thus far, this *outer* attention block captures long-range dependencies and global relational patterns with full nodes represented, serving as the primary backbone for feature propagation.

3.3 Local Topology-guided Iterative Edge Evolvolution

To support global attention with explicit structural reasoning, UECO interleaves a lightweight edge-evolving module *before* each attention layer. At a high level, inspired by the known convolutions [15],

$$\mathcal{F}_{\text{local}}(\mathbf{H}_r, \mathbf{H}_c, \mathbf{G} \mid \mathcal{I}) \to \mathbb{R}^{|\mathcal{V}|\times d} \times \mathbb{R}^{|\mathcal{V}|\times d} \times \mathbb{R}^{|\mathcal{E}|\times d} \tag{4}$$

is introduced to map and update the dual-view node semantics ($\mathbf{H}_r, \mathbf{H}_c$) and edge feature ($\mathbf{G}$) subject to a pair-wise adjacency indicator for a given instance ($\mathcal{I}$). For parameter efficiency, this module, with L_{local} layers, is defined outside and repeatedly reused across the L_{global} primary node-centric attention layers. Specifically, before the ℓ-th global attention layer, the local states are initialized as $\bar{\mathbf{h}}_i^{\ell,0} = \mathbf{h}_i^{\ell-1}$ and $\bar{\mathbf{g}}_{ij}^{\ell,0} = \mathbf{g}_{ij}^{\ell-1}$ ($\mathbf{H}^{\ell-1} = \{\mathbf{h}_i^{\ell-1}\}_{i\in\mathcal{V}}$, $\mathbf{G}^{\ell-1} = \{\mathbf{g}_{ij}^{\ell-1}\}_{(i,j)\in\mathcal{E}}$). For each inner local update step $t \in [L_{\text{local}}]$, we perform $\mathcal{F}_{\text{local}}^{(*)}(\mathbf{H}, \mathbf{G} \mid \mathcal{I}) \to \mathbb{R}^{|\mathcal{V}|\times d} \times \mathbb{R}^{|\mathcal{E}|\times d}$ on both feature modalities with respect to neighborhood topology and learnable $\mathbf{W}^A, \mathbf{W}^B, \mathbf{W}^C, \mathbf{W}^M, \mathbf{W}^N \in \mathbb{R}^{d\times d}$:

$$\tilde{\mathbf{g}}_{ij}^{\ell,t} = \mathbf{W}^A\bar{\mathbf{h}}_i^{\ell,t-1} + \mathbf{W}^B\bar{\mathbf{h}}_j^{\ell,t-1} + \mathbf{W}^C\bar{\mathbf{g}}_{ij}^{\ell,t-1},\ \ \bar{\mathbf{g}}_{ij}^{\ell,t} = \text{ReLU}\Big(\text{Norm}_{\text{edge}}(\tilde{\mathbf{g}}_{ij}^{\ell,t})\Big); \tag{5}$$

$$\tilde{\mathbf{h}}_i^{\ell,t} = \sum\nolimits_{(j,i)\in\mathcal{E}} \sigma(\bar{\mathbf{g}}_{ij}^{\ell,t}) \odot \mathbf{W}^M\bar{\mathbf{h}}_j^{\ell,t-1},\ \ \bar{\mathbf{h}}_i^{\ell,t} = \text{ReLU}\Big(\text{Norm}_{\text{node}}\left(\mathbf{W}^N\bar{\mathbf{h}}_i^{\ell,t-1} + \tilde{\mathbf{h}}_i^{\ell,t}\right)\Big). \tag{6}$$

To align with the dual-stream node representations, we in parallel perform $\mathcal{F}_{\text{local}}^{(r)}(\mathbf{H}_r, \mathbf{G} \mid \mathcal{I})$ and $\mathcal{F}_{\text{local}}^{(c)}(\mathbf{H}_c, \mathbf{G}^\top \mid \mathcal{I})$ for either source of node embeddings and combine the twin edge outputs with

$$\bar{\mathbf{G}}^{\ell,t} \leftarrow \text{ReLU}\left((\bar{\mathbf{G}}_r^{\ell,t} \,||\, \bar{\mathbf{G}}_c^{\ell,t})\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2, \tag{7}$$

with learnable $\mathbf{W}_1 \in \mathbb{R}^{2d\times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$ and biases, generating consistent edge embedding for the next layer. The merged edge output of the final layer, i.e., $\bar{\mathbf{G}}^{\ell,L_{\text{local}}}$, shall serve as the guiding effect in the mixture of the attention score (Eq. 2), which is introduced in detail in the next part.

3.4 Structure-Aware Attention Mixture

Algorithm 1: Interleaved Global-Local Encoding Pipeline in UECO

Input: Initial embeddings $\mathbf{H}_r^0, \mathbf{H}_c^0, \mathbf{G}^0$, graph instance $\mathcal{I}$

for $\ell = 1$ to L_{global} **do**

  $\bar{\mathbf{H}}_r^{\ell,L_{\text{local}}}, \bar{\mathbf{H}}_c^{\ell,L_{\text{local}}}, \bar{\mathbf{G}}^{\ell,L_{\text{local}}} \leftarrow \mathcal{F}_{\text{local}}\left(\mathbf{H}_r^{\ell-1}, \mathbf{H}_c^{\ell-1}, \mathbf{G}^{\ell-1} \mid \mathcal{I}\right)$

  $\mathbf{G}^{\ell} \leftarrow \bar{\mathbf{G}}^{\ell,L_{\text{local}}}$; $\mathbf{H}_r^{\ell}, \mathbf{H}_c^{\ell} \leftarrow \bar{\mathbf{H}}_r^{\ell,L_{\text{local}}}, \bar{\mathbf{H}}_c^{\ell,L_{\text{local}}}$

  $\mathbf{H}_r^{\ell}, \mathbf{H}_c^{\ell} \leftarrow \text{MixAtt}^{\ell}(\mathbf{H}_r^{\ell}, \mathbf{H}_c^{\ell}, \mathbf{G}^{\ell})$

end for

Among prior works, MatNet [13] pioneers the fusion of dot-product scores with asymmetric distance signals via an element-wise function for solving the ATSP (i.e., $\text{MixAtt} = f_\theta(\mathbf{E},\ \mathbf{Q}^\top\mathbf{K}/\sqrt{d_k})$), and later works like [23] resort to simpler additive ways ($\text{MixAtt} = \lambda\mathbf{E} + \mathbf{Q}^\top\mathbf{K}/\sqrt{d_k}$), etc. We now instantiate $\text{MixAtt}(\cdot)$ in Eq. 2 which learns a dynamic mixture of content-based and structure-based scores. The key is to parameterize a learned mixture between i) the node-centric similarity induced by $(\mathbf{Q}_i, \mathbf{K}_i)$ and ii) an edge-induced structural relevance distilled from the evolved edge embedding $\bar{\mathbf{G}}^{\ell,L_{\text{local}}}$ returned by $\mathcal{F}_{\text{local}}$. This yields a dynamic score function whose structural component is *iteratively refined* across layers, rather than injected in a static one-shot manner using the untuned edge feature from the raw input.

For head $i \in [h]$, let $\mathbf{G} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times d}$ be the (dense) edge embedding used to guide attention at the current depth. We first compute the conventional scaled dot-product score:

$$\mathbf{S}_{\text{dot}}^{(i)} = \frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_h}} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}. \tag{8}$$

In parallel, we project the shared edge embedding to a per-head structure-aware relevance score.

$$\mathbf{S}_{\text{str}} = \mathbf{G}\mathbf{W}_{\text{edge}},\ \ \mathbf{S}_{\text{str}}^{(i)} = \mathbf{S}_{\text{str}}[:,:,i] \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}, \tag{9}$$

with learnable $\mathbf{W}_{\text{edge}} \in \mathbb{R}^{d\times h}$. Intuitively, $\mathbf{S}_{\text{str}}^{(i)}$ summarizes, for head i, which pairwise relations are structurally salient under the current local topology encoding. We then concatenate the two scores along the feature dimension so each pair-wise entry (i.e., edge) (u, v) carries a 2D feature $(\mathbf{S}_{\text{dot}}^{(i)}(u,v), \mathbf{S}_{\text{str}}^{(i)}(u,v))$ which are subsequently fused via a lightweight MLP applied point-wise:

$$\mathbf{S}_{\text{mix}}^{(i)} = \text{ReLU}\Big((\mathbf{S}_{\text{dot}}^{(i)} \,||\, \mathbf{S}_{\text{str}}^{(i)})\mathbf{W}_1 + \mathbf{b}_1\Big)\,\mathbf{w}_2 + \mathbf{b}_2, \tag{10}$$

where $\mathbf{W}_1 \in \mathbb{R}^{2\times d_{\text{ms}}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{\text{ms}}}$, $\mathbf{w}_2 \in \mathbb{R}^{d_{\text{ms}}\times 1}$, and $\mathbf{b}_2 \in \mathbb{R}$ are learnable parameters, and d_{ms} is the hidden width of the mixture MLP, ensuring $\mathbf{S}_{\text{mix}}^{(i)} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. Note that Eq. 10 is not merely an additive correction: it enables head-specific, nonlinear interactions between global similarity and local structural cues. Finally, the attention weights and head output are obtained: $\mathbf{A}^{(i)} = \text{softmax}(\mathbf{S}_{\text{mix}}^{(i)})$ and $\mathbf{head}_i = \mathbf{A}^{(i)}\mathbf{V}_i$, where $\text{softmax}(\cdot)$ is row-wise and $\mathbf{head}_i \in \mathbb{R}^{|\mathcal{V}|\times d_h}$. Substituting into Eq. 2, the resulting MHA output is then passed through the residual-normalization-FFN stack in Sec. 3.2.

Remark. Since $\mathbf{S}_{\text{str}}^{(i)}$ is computed from $\mathbf{G}$, and $\mathbf{G}$ is *iteratively evolved* by $\mathcal{F}_{\text{local}}$, the structural-signal-modulated attention is dynamic and depth-adaptive. In this way, UECO realizes a closed-loop coupling: local topology updates inform global attention via $\mathbf{S}_{\text{str}}^{(i)}$, while global node semantics feed back to the next edge evolution through Eq. 4.

3.5 Output Compatibility with Paradigm-Specific Decoders

UECO is designed to be orthogonal to downstream decision paradigms as well as learning objectives. Thus, the representations produced by the encoder can be consumed by different decoding strategies via simple output adapters without architectural modifications. **For GP and AE methods** (i.e., heatmap-based, often trained supervised), we project the final edge or node (merged from row and column views) embeddings to a two-channel logit map via a lightweight 1×1 convolutional head,

$$\mathbf{Z}_{\text{GP}}^{edge} = \text{Conv2d}(\mathbf{G}^{L_{\text{global}}}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times 2}, \quad \mathbf{Z}_{\text{GP}}^{node} = \text{Conv2d}\left(\text{MLP}\left(\mathbf{H}_r^{L_{\text{global}}}, \mathbf{H}_c^{L_{\text{global}}}\right)\right) \in \mathbb{R}^{|\mathcal{V}|\times 2}, \tag{11}$$

which encodes the likelihood of each decision variable being selected (edge-wise for tasks such as TSP, and node-wise for node-oriented problems like MIS). These logits are typically used for computing a binary cross entropy loss against the off-the-shelf optimal labels to conduct supervised learning and, at the solving stage, support classical greedy construction or task-specific decoders. **For LC methods** (Markov modeled, often trained reinforced), dual encoded node outputs $\mathbf{H}_r^L, \mathbf{H}_c^L$ are naturally fed to a

Table 1: Comparison with representative edge-/matrix-aware attention works for NCO. "Dyn. edge" indicates whether learned edge states are updated across network depth. "L-G loop" denotes repeated interleaving between local edge/topology updates and global node-centric attention. "X-para." denotes cross-paradigm generality. "Para. / Obj." denotes paradigm and learning objective.

Method	Input	Dyn. edge	L-G loop	Multi-input	X-para.	Para. / Obj.	Evaluated tasks
MatNet [13]	Edge	×	×	×	×	LC-RL	TSP, ATSP, FFSP
GREAT [27]	Edge	✓	×	×	△	LC-RL, GP-SL	TSP, ATSP
UniCO [24]	Edge	△	×	×	△	LC-RL, GP-SL	TSP, ATSP
EFormer [28]	Edge / Node	△	×	△	×	LC-RL	TSP, ATSP, CVRP
COExpander [21]	Edge / Node / Hybrid	×	×	✓	△	GP-SL, AE-SL	TSP, ATSP, MIS, MCl, MCut
UniteFormer [29]	Edge / Node / Hybrid	△	×	✓	×	LC-RL	TSP, ATSP, CVRP
UECO (ours)	Edge / Node / Hybrid	✓	✓	✓	✓	LC-RL, GP-SL, AE-SL	TSP, ATSP, CVRP, ACVRP, MIS, MCl, MCut, KP, FFSP

Notation. ✓ means explicitly supported by the main design or main experiments; × means not the main design target; △ denotes partial, appendix-level, solver-dependent, or module-local support. More detailed comparison is in Appendix C.2.

MatNet-style autoregressive decoder. With the attention-based intermediate $\mathbf{Q}_{\text{dec}} = \mathbf{H}_r\mathbf{W}^Q_{\text{dec}}, \mathbf{K}_{\text{dec}} = \mathbf{H}_c\mathbf{W}^K_{\text{dec}}, \mathbf{V}_{\text{dec}} = \mathbf{H}_c\mathbf{W}^V_{\text{dec}}$ and $\mathbf{o}^{n+1}_{\text{dec}} = (\text{softmax}(\mathbf{Inf}_{\mathbf{s}[n'\le n]} + \mathbf{q}^{n\top}_{\text{dec}}\mathbf{K}_{\text{dec}}/\sqrt{d})\mathbf{V}_{\text{dec}})\mathbf{W}^O_{\text{dec}}$, we have

$$\mathbf{z}^{n+1}_{\text{LC}} = \text{softmax}\left(\mathbf{Inf}_{\mathbf{s}[n'\le n]} + \tanh\left(\frac{\mathbf{H}^\top_c\mathbf{o}^{n+1}_{\text{dec}}}{\sqrt{d}}\right)\right) \in \mathbb{R}^{|\mathcal{V}|}, \tag{12}$$

predicting the next node selection conditioned on the current state and a constraint-aware mask (infeasible actions set to $\mathbf{Inf}$), supporting practical reinforcement policy learning.

4 Related Work

Neural Combinatorial Optimization. Learning-based CO solvers can be broadly categorized into end-to-end paradigms, i.e., local construction (LC), global prediction (GP), adaptive expansion (AE), and improvement-based, neural-heuristic, or divide-and-conquer (D&C) frameworks. LC methods [10–14, 30–32] construct solutions by auto-regressively selecting variables with a Transformer-style decoder while maintaining feasibility. GP methods [15, 16, 33–38] instead generate probability heatmaps in a single forward pass and rely on simple decoders (e.g., greedy) to enforce feasibility. AE approaches [19–21] interpolate between these extremes by adaptively adjusting decision granularity during decoding. Beyond these paradigms, improvement-based methods [23, 39–41] learn to refine existing solutions through local rewrites, while neural-heuristic hybrids [42–47] guide or parameterize classical solvers with neural models. D&C methods [48–51] split large COPs into smaller subproblems and address them via existing, orthogonal medium-scale local neural solvers.

(Unified) Backbone Designs for NCO. Earlier works generally resort to either the established Transformer [26, 52, 53] or GCN [15, 18, 54]. The former only take node coordinates as input and learns node embeddings, whereas the latter iterates both node and edge modalities via local topological aggregation. Beyond them, GREAT [27] and EFormer [28] propose edge-centric Transformers, and MatNet [13] attempts to fuse edge weights into the attention score element-wise. More recently, GOAL [32] and UniCO [24] explore multi-task CO solving with task-specific headers or problem reformulation. UnifyML4TSP [55] recomposes the backbones within learning-based TSP solving, OPTFM [56] introduces a multi-view graph Transformer for modeling Mixed Integer Linear Programs (MILPs), and UniteFormer [29] further integrates the node and edge modalities in Transformers within Vehicle Routing Problems (VRPs). Table 1 gives an in-depth multi-dimensional comparison.

5 Experiments

Evaluated Problems and Datasets. To cover graph-based COPs with different focuses of decision variables and different levels of constraint complexity, we assess our model on a broader spectrum of tasks (than prior norms [33, 36, 57]): the Traveling Salesman Problem (TSP), Asymmetric TSP (ATSP), Capacitated Vehicle Routing Problem (CVRP), Asymmetric CVRP (ACVRP), Maximum Independent Set (MIS), Maximum Clique (MCl), and Maximum Cut (MCut). We adopt exactly the same training and testing datasets open-sourced by [57] for reproducible comparison. For **in-distribution** test, we experiment, by convention, on uniform graphs ($\mathbf{x} \sim [0, 1]^2$) for TSP/CVRP, t-mat class ($\mathbf{e} \sim [0, 1]$ with triangle inequality holding) instances [13, 24, 58] for ATSP/ACVRP,

Table 2: Results on TSP. G: greedy, 2OPT: optional 2-Opt iterative refinement; OS: One-shot forward prediction, Gen: diffusion-based generative modeling; Trans.: (node-centric) graph Transformer, Uni.: our UECO; **D**: decision rounds for AE paradigm, **I**: inference steps for generative methods. Full results in Appendix E.1.

Method	Category	**TSP-50** (1280 inst.)			**TSP-100** (1280 inst.)			**TSP-500** (128 inst.)			**TSP-1000** (128 inst.)		
		Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
Concorde	Exact	5.69	0.000%	0.06s	7.76	0.000%	0.24s	16.55	0.000%	18.67s	23.12	0.000%	84.41s
LKH-3 (trails=500)	Heuristic	5.69	0.001%	0.06s	7.76	0.001%	0.18s	16.55	0.002%	1.85s	23.13	0.005%	4.64s
UTSP (G)	GP-UL(OS)-GCN	7.10	24.808%	0.01s	9.38	20.909%	0.01s	19.76	19.435%	0.06s	N/A	N/A	N/A
DIMES (G)	GP-RL(OS)-GCN	6.33	11.233%	0.00s	8.76	12.918%	0.01s	18.89	14.155%	0.27s	26.82	16.013%	0.61s
GNNGLS (G)	GP-SL(OS)-GCN	6.18	8.651%	0.02s	8.97	15.597%	0.02s	–	–	–	–	–	–
GCN4CO (G)	GP-SL(OS)-GCN	5.72	0.520%	0.01s	7.94	2.334%	0.01s	18.88	14.098%	0.05s	28.02	21.188%	0.20s
GCN4CO (G + 2OPT)	GP-SL(OS)-GCN	5.69	0.066%	0.01s	7.78	0.243%	0.01s	16.85	1.863%	0.06s	23.97	3.703%	0.23s
UECO (G)	GP-SL(OS)-Uni.	5.70	0.147%	0.01s	7.81	0.635%	0.01s	18.24	10.259%	0.06s	26.61	15.125%	0.21s
UECO (G + 2OPT)	GP-SL(OS)-Uni.	**5.69**	**0.034%**	0.01s	**7.76**	**0.091%**	0.01s	**16.80**	**1.559%**	0.06s	**23.63**	**2.207%**	0.23s
Fast-T2T (G, I=1)	GP-SL(Gen)-GCN	5.72	0.612%	0.01s	7.94	2.381%	0.02s	18.72	13.128%	0.07s	26.78	15.854%	0.30s
Fast-T2T (G, I=5)	GP-SL(Gen)-GCN	5.69	0.061%	0.03s	7.77	0.230%	0.06s	18.58	12.280%	0.14s	26.57	14.918%	0.57s
Fast-T2T (G + 2OPT, I=5)	GP-SL(Gen)-GCN	5.69	0.029%	0.03s	7.76	0.078%	0.06s	16.86	1.903%	0.16s	23.67	2.393%	0.61s
UECO (G, I=1)	GP-SL(Gen)-Uni.	5.71	0.397%	0.01s	7.89	1.731%	0.02s	18.53	11.965%	0.07s	26.74	15.656%	0.23s
UECO (G, I=5)	GP-SL(Gen)-Uni.	5.69	0.026%	0.06s	7.77	0.140%	0.06s	18.30	10.615%	0.20s	26.55	14.766%	0.68s
UECO (G + 2OPT, I=5)	GP-SL(Gen)-Uni.	**5.69**	**0.014%**	0.06s	**7.76**	**0.053%**	0.06s	**16.79**	**1.489%**	0.21s	**23.55**	**1.845%**	0.69s
GOAL (G)	LC-SL(OS)-Trans.	5.74	0.965%	0.39s	7.91	1.920%	0.76s	18.01	8.825%	3.96s	25.38	9.789%	10.34s
AM (G)	LC-RL(OS)-Trans.	5.75	1.040%	0.01s	7.95	2.520%	0.03s	–	–	–	–	–	–
Sym-NCO (G)	LC-RL(OS)-Trans.	5.70	0.258%	0.05s	7.87	1.491%	0.10s	–	–	–	–	–	–
PreferenceCO (G)	LC-RL(OS)-Trans.	5.70	0.140%	0.03s	7.77	0.196%	0.01s	–	–	–	–	–	–
UniteFormer-XE (G)	LC-RL(OS)-Trans.	5.76	1.313%	0.22s	8.11	4.600%	0.43s	–	–	–	–	–	–
POMO (G)	LC-RL(OS)-Trans.	5.70	0.196%	0.04s	7.83	0.986%	0.05s	18.19	9.913%	0.23s	27.39	18.483%	1.14s
POMO (G + 2OPT)	LC-RL(OS)-Trans.	5.70	0.159%	0.04s	7.82	0.798%	0.05s	17.58	6.255%	0.55s	25.30	9.458%	6.13s
UECO (G)	LC-RL(OS)-Uni.	5.69	0.039%	0.06s	7.77	0.216%	0.06s	16.82	1.632%	0.35s	24.21	4.736%	1.27s
UECO (G + 2OPT)	LC-RL(OS)-Uni.	**5.69**	**0.035%**	0.06s	**7.77**	**0.193%**	0.06s	**16.78**	**1.445%**	0.42s	**23.86**	**3.192%**	2.95s
COExpander (G, D=3)	AE-SL(Gen)-GCN	5.70	0.244%	0.02s	7.85	1.164%	0.03s	18.77	13.447%	0.13s	26.81	15.966%	0.45s
COExpander (G + 2OPT, D=3)	AE-SL(Gen)-GCN	5.69	0.029%	0.02s	7.77	0.124%	0.03s	16.86	1.897%	0.15s	23.63	2.198%	0.50s
UECO (G, D=3)	AE-SL(Gen)-Uni.	5.69	0.089%	0.04s	7.79	0.411%	0.04s	18.53	11.989%	0.14s	26.76	15.763%	0.45s
UECO (G + 2OPT, D=3)	AE-SL(Gen)-Uni.	**5.69**	**0.014%**	0.04s	**7.76**	**0.066%**	0.04s	**16.77**	**1.330%**	0.15s	**23.53**	**1.773%**	0.49s

Table 3: Main results on MIS.

Method	Category	**RB-[200-300]** (500 inst.)			**RB-[800-1200]** (500 inst.)			**ER-[700-800]** (128 inst.)			**SATLIB** (500 inst.)		
		Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time
KaMIS (60s)	Heuristic	20.09	0.000%	45.81s	43.00	0.000%	56.97s	44.97	0.000%	60.75s	425.95	0.000%	24.37s
Gurobi (60s)	OR	20.09	0.000%	0.54s	42.19	1.829%	33.84s	38.78	13.749%	60.49s	425.92	0.007%	3.95s
VAG-CO (G)	GP-UL(OS)-GCN	17.31	13.709%	0.016s	34.00	20.779%	0.03s	33.81	24.804%	0.03s	–	–	–
Meta-EGN (G)	GP-UL(OS)-GCN	17.69	11.778%	0.08s	34.67	19.259%	0.13s	33.15	26.280%	0.12s	412.12	3.250%	0.24s
GCN4CO (G)	GP-SL(OS)-GCN	18.17	9.413%	0.01s	36.06	16.036%	0.05s	35.85	20.267%	0.04s	421.06	1.154%	0.02s
UECO (G)	GP-SL(OS)-Uni.	**18.53**	**7.675%**	0.01s	**36.30**	**15.830%**	0.04s	**36.47**	**18.895%**	0.04s	**422.88**	**0.725%**	0.02s
DiffUCO (G)	GP-UL(Gen)-GCN	19.20	4.369%	0.47s	38.49	10.428%	4.71s	–	–	–	–	–	–
Fast-T2T (G, I=1)	GP-SL(Gen)-GCN	18.40	8.305%	0.01s	36.39	15.219%	0.04s	35.71	20.580%	0.04s	421.58	1.031%	0.02s
Fast-T2T (G, I=20)	GP-SL(Gen)-GCN	19.33	3.710%	0.19s	38.94	9.388%	0.71s	40.20	10.618%	0.64s	424.56	0.329%	0.19s
UECO (G, I=1)	GP-SL(Gen)-Uni.	18.62	7.276%	0.01s	36.71	14.524%	0.04s	36.40	19.049%	0.04s	422.87	0.729%	0.02s
UECO (G, I=20)	GP-SL(Gen)-Uni.	**19.48**	**2.999%**	0.19s	**39.10**	**8.999%**	0.63s	**40.27**	**10.459%**	0.57s	**424.94**	**0.238%**	0.21s
LwD (G)	AE-RL(OS)-GCN	–	–	–	–	–	–	37.29	17.068%	0.59s	–	–	–
COExpander (G, D=20)	AE-SL(Gen)-GCN	19.66	2.088%	0.11s	41.23	4.060%	0.62s	42.38	3.746%	0.47s	425.05	0.215%	0.23s
UECO (G, D=20)	AE-SL(Gen)-Uni.	**19.83**	**1.262%**	0.13s	**41.79**	**2.775%**	0.57s	**42.73**	**4.961%**	0.42s	**425.31**	**0.153%**	0.22s

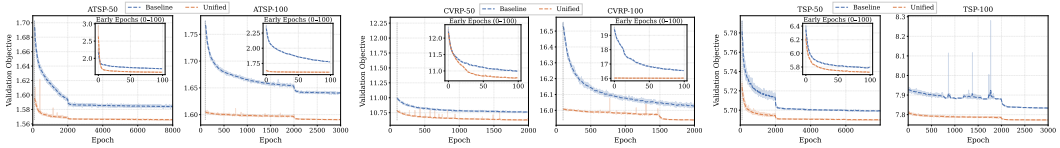
RB [59] and ER [60] graphs for MIS/MCl, and BA [61] graphs for MCut. For **generalization** test, we report results extensively across TSPLIB, CVRPLIB, SATLIB, TWITTER [62], and COLLAB [63].

Compared Baselines. To obtain the reference solutions for computing the optimality gaps, we employ exact or heuristic solvers, i.e., Concorde [64] for TSP, LKH [65] for ATSP, KaMIS [66] for MIS, and Gurobi [67] for MCl and MCut. For NCO approaches, we compare our UECO with UTSP [68], DIMES [33], GNNGLS [69], VAG-CO [34], Meta-EGN [35], and GCN4CO [15] (re-implemented version upon [57] with significant performance gain) for the GP paradigm with one-shot (OS) prediction; Fast-T2T [37] and DiffUCO [36] for GP with generative (Gen) multiple inference steps; AM [10], MatNet [13], POMO [11], Sym-NCO [12], and the latest GOAL [32], MatPOENet [24], UniteFormer [29] and PreferenceCO [70] for the LC paradigm (mostly RL-driven); LwD [19] and COExpander [21] for the AE paradigm. Also, we adapt UECO as local solvers for D&C frameworks [20, 49, 51] for tackling extremely large routing instances. We further contextualize our aligned evaluation scope and benchmark selection within community consensus in Appendix C.

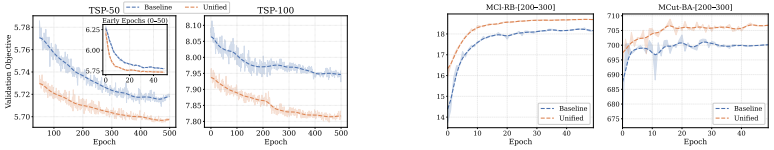
Metrics and Protocol. By convention, we report the absolute objective value (**Obj.**), the relative optimality gap (**Gap**), and per-instance average solving time (**Time**) on every benchmark. We adopt the pre-trained weights for baselines where publicly available, otherwise retrain them with the official settings. A single thread, single batch-size greedy decoder is strictly enforced with optional 2-OPT or classic local search (LS) refinement upon the consistent and open-source Python interface from ML4CO-Kit for fairness. The hardware, hyper-parameters and settings are specified in Appendix D.

5.1 Main Results with Plug-and-Play Ablations

We first report the in-distribution performance on seven COPs in Tables 2, 3, 4, 5, 9, and 12. We train multiple variants of UECO under exactly the same experimental settings as their corresponding baselines spanning diverse problem benchmarks, decision paradigms and learning objectives, with only the architecture replaced by UECO. Notably, UECO consistently achieves superior solution



(a) ATSP. Baseline: MatNet (LC-RL). (b) CVRP. Baseline: POMO (LC-RL). (c) TSP. Baseline: POMO (LC-RL).



(d) TSP. Baseline: GCN4CO (GP-SL). (e) MCl/MCut. Baseline: COExpander (AE-SL).

Figure 3: Training curves across five settings. Replacing only the encoder in respective baselines with UECO.

Table 4: Results on CVRP. Time limit for HGS: 1/20/60/360s for CVRP-50/100/200/500. **S**: number of parallel samples. # denotes results quoted from the original papers.

Method	Category	CVRP-50 (10000 inst.)			CVRP-100 (10000 inst.)			CVRP-200 (100 inst.)			CVRP-500 (100 inst.)		
		Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
HGS	Heuristic	10.37	0.000%	1.01s	15.56	0.000%	20.03s	19.63	0.000%	60.02s	37.15	0.000%	360.38s
Fast-T2T (G, I=1)	GP-SL(Gen)-GCN	12.64	21.835%	0.01s	19.20	23.333%	0.01s	25.06	27.616%	0.06s	47.75	28.509%	0.09s
COExpander (G, D=3)	AE-SL(Gen)-GCN	11.98	15.407%	0.03s	17.50	12.343%	0.05s	22.40	13.977%	0.15s	43.90	18.199%	0.55s
AM# (G)	LC-RL(OS)-Trans.	10.98	5.86%	–	16.80	7.34%	–	–	–	–	–	–	–
POMO# (G)	LC-RL(OS)-Trans.	10.74	3.52%	0.08s	16.15	3.00%	0.16s	–	–	–	–	–	–
PreferenceCO (G)	LC-RL(OS)-Trans.	10.57	2.001%	0.01s	15.84	1.772%	0.02s	–	–	–	–	–	–
UniteFormer-XE (G)	LC-RL(OS)-Trans.	10.64	2.653%	0.31s	16.35	5.057%	0.53s	–	–	–	–	–	–
UniteFormer-XE (S=8)	LC-RL(OS)-Trans.	10.47	1.042%	0.32s	16.00	2.781%	0.58s	–	–	–	–	–	–
Sym-NCO (G)	LC-RL(OS)-Trans.	10.77	3.891%	0.09s	16.22	4.241%	0.17s	20.66	5.274%	0.32s	40.38	8.723%	0.77s
GOAL (G)	LC-SL(OS)-Trans.	10.91	5.193%	0.50s	16.34	5.005%	0.96s	21.44	9.263%	1.98s	38.85	4.587%	7.10s
GOAL (S=16)	LC-SL(OS)-Trans.	10.62	2.444%	0.55s	15.99	2.743%	1.28s	20.67	5.272%	5.07s	38.28	3.051%	64.43s
GOAL (S=16 + LS)	LC-SL(OS)-Trans.	10.53	1.513%	0.55s	15.84	1.757%	1.29s	20.20	2.882%	5.10s	37.90	2.031%	64.59s
UECO (G)	LC-RL(OS)-Uni.	10.55	1.783%	0.03s	15.90	2.156%	0.06s	20.17	2.788%	0.12s	38.35	3.250%	0.30s
UECO (S=16)	LC-RL(OS)-Uni.	10.45	0.836%	0.04s	15.78	1.397%	0.08s	20.05	2.171%	0.18s	38.22	2.891%	1.35s
UECO (S=16 + LS)	LC-RL(OS)-Uni.	**10.41**	**0.452%**	0.04s	**15.70**	**0.859%**	0.08s	**19.91**	**1.412%**	0.19s	**37.83**	**1.842%**	1.43s

Table 5: Results on ATSP. # denotes results quoted from the original paper.

Method	Category	ATSP-50 (2500 inst.)			ATSP-100 (2500 inst.)			ATSP-200 (100 inst.)			ATSP-500 (100 inst.)		
		Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
LKH-3 (trails=1000)	Heuristic	1.55	0.000%	0.10s	1.57	0.000%	0.24s	1.57	0.000%	0.72s	1.57	0.000%	4.38s
GCN4CO (G)	GP-SL(OS)-GCN	1.67	7.261%	0.01s	1.67	6.408%	0.01s	1.67	6.777%	0.05s	1.73	9.880%	0.16s
Fast-T2T (G, I=1)	GP-SL(Gen)-GCN	1.66	6.948%	0.01s	1.67	6.329%	0.01s	1.68	7.280%	0.04s	1.68	6.443%	0.13s
COExpander (G, D=3)	AE-SL(Gen)-GCN	1.64	5.304%	0.02s	1.65	5.086%	0.02s	1.63	4.078%	0.10s	1.65	4.634%	0.40s
GOAL (G)	LC-SL(OS)-Trans.	1.60	2.567%	0.40s	1.64	5.010%	0.72s	1.64	4.956%	1.56s	1.66	5.201%	6.33s
MatPOENet (G)	LC-RL(OS)-Trans.	1.58	1.601%	0.04s	1.63	3.768%	0.06s	–	–	–	–	–	–
UniteFormer# (G)	LC-RL(OS)-Trans.	–	0.917%	–	–	3.170%	–	–	–	–	–	–	–
MatNet (G)	LC-RL(OS)-Trans.	1.58	1.373%	0.03s	1.62	3.456%	0.03s	–	–	–	–	–	–
MatNet (S=16)	LC-RL(OS)-Trans.	1.56	0.334%	0.04s	1.59	1.653%	0.06s	–	–	–	–	–	–
MatNet (S=16 + 2OPT)	LC-RL(OS)-Trans.	1.56	0.319%	0.04s	1.59	1.586%	0.06s	–	–	–	–	–	–
UECO (G)	LC-RL(OS)-Uni.	1.56	0.656%	0.02s	1.59	1.365%	0.03s	1.60	2.276%	0.08s	1.63	3.604%	0.18s
UECO (S=16)	LC-RL(OS)-Uni.	**1.56**	**0.328%**	0.02s	**1.58**	**0.663%**	0.07s	**1.59**	**1.414%**	0.24s	**1.62**	**2.920%**	1.93s
UECO (S=16 + 2OPT)	LC-RL(OS)-Uni.	**1.56**	**0.317%**	0.02s	**1.58**	**0.646%**	0.07s	**1.59**	**1.373%**	0.24s	**1.62**	**2.844%**	2.01s

Table 6: Generalization results on CVRPLIB (22 inst., following [31, 70, 73]).

Method	Gap↓
POMO [11]	5.26%
Sym-NCO [12]	9.99%
LEHD [52]	11.11%
BQ-NCO [31]	10.60%
Omni-POMO [74]	5.04%
ELG [73]	4.51%
UECO (ours)	**4.46%**

Table 7: Generalization results on TSPLIB (29 inst., following [17, 37, 69]).

Method	Gap↓
AM [10]	16.77%
POMO [11]	3.20%
Learn2OPT [75]	1.73%
GNNGLS [69]	1.53%
DIFUSCO [76]	0.97%
Fast-T2T [37]	0.28%
UECO (ours)	**0.19%**

quality while maintaining competitive solving efficiency. Further, Figs. 3c–3e illustrate the training dynamics which show stable performance gains throughout training across heterogeneous settings. Beyond the rich spectrum of baselines, we also provide **additional comparisons and discussion** in Appendix E.2, where UECO outperforms our best-effort reproductions of L2C [71] and INViT [72].

5.2 Generalization Study and Real-world Evaluation

Case 1: Cross-scale Generalization. First, the TSP-1000 results reported in Table 2 are obtained using models trained on TSP-100. Further, Table 11 evaluates MIS/MCl/MCut models trained on 800–1200-node graphs and tested on instances with up to 11000 nodes [77–79]. Across all settings, UECO shows strong extrapolation ability, outperforming most recent unsupervised [35, 36] and generative [37, 76] approaches. On MCut-[2000-3000] and MIS-[1400-1600], UECO even surpasses Gurobi under a 3600s time limit while requiring less than 1s and 5s per instance, respectively.

Table 8: UECO as plug-in sub-solvers in D&C frameworks for ultra-large TSP instances. Some results are from [51].

Methods	TSP-1K		TSP-10K		TSP-100K	
	Obj.↓	Gap↓	Obj.↓	Gap↓	Obj.↓	Gap↓
LKH	23.12	–	71.75	–	226.26	–
DIMES+S	26.36	14.01%	86.3	20.18%	286.1	26.37%
GLOP (more)	23.84	3.12%	75.3	4.90%	238.0	5.12%
UDC-$x_{2/50}$	23.79	2.92%	74.7	4.11%	235.8	4.15%
UDC-x_{250}	–	–	74.7	4.03%	235.6	4.05%
DualOPT-AM	23.74	2.70%	73.18	1.99%	230.13	1.71%
DualOPT-UECO	**23.27**	**0.66%**	**72.21**	**0.64%**	**229.76**	**1.55%**

Case 2: Cross-distribution Generalization. We first test models trained on uniform TSP/ATSP data on out-of-distribution (OoD) instances drawn from Gaussian, clustered, and binary Hamiltonian distributions, following M^{2}GenCO [80]. For node tasks, models trained on MIS/MCl-RB and MCut-BA graphs are evaluated on unseen HK [81] graphs. Due to space limitations, we defer these results to Table 21 in Appendix E.4.

Table 10: Results on TWITTER (195 inst.) and COLLAB (1k inst.) for MCl.

Method	TWITTER		COLLAB	
	Obj.↑	Time	Obj.↑	Time
Gurobi	14.21	0.28s	42.11	0.06s
COExpander	13.53	0.07s	41.87	0.14s
Meta-EGN	13.68	0.13s	41.99	0.13s
UECO (ours)	**13.93**	**0.03s**	**42.07**	**0.02s**

Table 9: Results on MCl and MCut (500 inst. each). Gurobi time limit: 60s/360s for [200-300]/[800-1200] data.

Method	Category	MCl-RB-[200-300]			MCl-RB-[800-1200]			MCut-BA-[200-300]			MCut-BA-[800-1200]		
		Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time
Gurobi	OR	19.08	0.000%	0.90s	40.18	0.000%	276.66s	727.84	0.000%	60.61s	2936.89	0.000%	300.21s
VAG-CO (G)	GP-UL(OS)-GCN	15.08	20.720%	0.02s	29.05	26.604%	0.03s	700.97	3.706%	0.02s	2884.09	1.815%	0.02s
Meta-EGN (G)	GP-UL(OS)-GCN	17.86	6.400%	0.07s	34.80	13.180%	0.10s	672.75	7.563%	0.06s	2748.31	6.425%	0.09s
GCN4CO (G)	GP-SL(OS)-GCN	18.08	5.782%	0.01s	**38.62**	**4.331%**	0.04s	705.33	3.102%	0.01s	2827.48	3.744%	0.02s
UECO (G)	GP-SL(OS)-Uni.	**18.53**	**3.346%**	0.01s	37.59	7.192%	0.04s	**709.47**	**2.539%**	0.01s	**2834.83**	**3.494%**	0.02s
Fast-T2T (G, I=1)	GP-SL(Gen)-GCN	18.26	4.870%	0.02s	36.40	9.809%	0.04s	702.38	3.504%	0.01s	2783.83	5.231%	0.02s
Fast-T2T (G, I=20)	GP-SL(Gen)-GCN	18.61	2.850%	0.20s	35.38	12.672%	0.71s	725.62	0.319%	0.17s	2948.11	-0.369%	0.19s
UECO (G, I=1)	GP-SL(Gen)-Uni.	18.65	2.607%	0.01s	38.14	5.758%	0.04s	706.58	2.931%	0.01s	2836.56	3.431%	0.02s
UECO (G, I=20)	GP-SL(Gen)-Uni.	**18.88**	**1.218%**	0.19s	**38.20**	**5.544%**	0.63s	**726.39**	**0.213%**	0.17s	2944.97	-0.266%	0.20s
COExpander (G, D=20)	AE-SL(Gen)-GCN	18.77	1.892%	0.05s	36.75	8.817%	0.15s	727.53	0.058%	0.18s	2978.20	-1.387%	0.20s
UECO (G, D=20)	AE-SL(Gen)-Uni.	**18.92**	**1.036%**	0.05s	**38.92**	**3.480%**	0.13s	**727.86**	**0.014%**	0.18s	**2981.70**	**-1.508%**	0.20s

Table 11: Generalization results on ultra-large node-decision problems.

Method	MCl-RB-[2K-3K]		MCut-BA-[2K-3K]		MIS-RB-[2K-3K]		MIS-ER-[1400-1600]		MIS-ER-[9K-11K]	
	Obj.↑	Time	Obj.↑	Time	Obj.↑	Time	Obj.↑	Time	Obj.↑	Time
Gurobi	81.52	3606.20s	7217.90	3601.34s	48.56	3426.21s	44.81	3602.52s	–	–
KaMIS	–	–	–	–	49.26	60.96s	50.94	60.82s	381.44	623.31s
GCN4CO	58.44	0.22s	6979.26	0.06s	38.76	0.24s	37.68	0.12s	296.00	1.73s
VAG-CO	55.58	0.15s	7124.88	0.06s	37.26	0.15s	36.91	0.07s	287.38	1.27s
GOAL	57.12	32.68s	–	–	42.64	24.50s	45.59	9.97s	OOM	
Meta-EGN	77.16	0.68s	6712.12	0.54s	38.84	0.74s	37.56	0.27s	273.06	7.17s
Fast-T2T	60.86	0.24s	7372.10	0.72s	38.38	8.98s	44.09	5.12s	305.94	5.64s
COExpander	74.92	3.39s	7369.98	0.72s	46.64	14.66s	**48.43**	4.52s	362.38	20.14s
UECO	**79.68**	0.34s	**7377.26**	0.68s	**47.46**	5.19s	47.89	4.54s	**364.58**	18.87s

Table 12: Main results on ACVRP-200. More results and analysis in Appendix E.3.

Method	Obj.↓	Gap↓
LKH-10K	2.165	0.00%
HGS-Long	1.973	-8.83%
MatNet [13]	2.237	3.36%
ReLD [82]	2.164	-0.05%
RADAR [58]	2.148	-0.75%
UECO	**2.101**	**-2.94%**

Case 3: Synergy with Divide-and-Conquer Frameworks. D&C has become a common strategy for ultra-large routing by applying local NCO solvers to tractable sub-problems. Using DualOPT [20] as the D&C backbone, we replace its sub-solvers with UECO-LC pretrained on TSP-{50, 100, 500}. In Table 8, DualOPT-UECO achieves stronger results up to TSP-100K, showing that UECO transfers effectively as a plug-in local solver in scalable D&C pipelines. The test data follow [20, 49] which contain 128, 16, and 1 instances for TSP-1K/10K/100K respectively. Full results are in Appendix E.5.

Case 4: Results on Real-world Benchmarks. a) On CVRPLIB and TSPLIB. Following [37, 69, 70], in Tables 6 and 7, we report results on 22 selected CVRPLIB-Set-X and 29 TSPLIB instances, using models trained on uniform TSP/CVRP-100. Full *per-instance* results are in Appendix E.6. **b) On TWITTER and COLLAB.** We report in Table 10 the results for MCl on the two famous datasets from social networks, where UECO remains consistently competitive against strong neural baselines.

5.3 Supplementary Studies and Analyses

RQ1: Effect of Scaling Model Parameters. As shown in Figs. 4 and 5, increasing the number of global attention layers, local topology layers, or enlarging the hidden dimension leads to steady gains, particularly under GP, validating the synergy among the interleaved edge evolution in refining structural information for global mapping, and that UECO scales reliably with the model capacity.

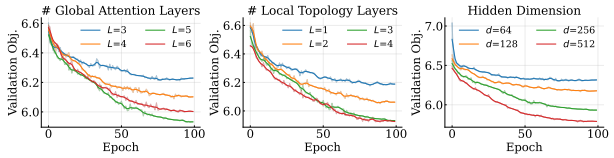


Figure 4: Ablation on architectural parameters (TSP, GP-SL).

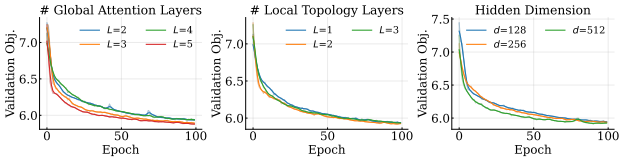


Figure 5: Ablation on architectural parameters (TSP, LC-RL).

RQ2: Effect of Decision Rounds and Inference Steps in Generative Modeling. Fig. 6 examines the effect of $\mathbf{D}$ and $\mathbf{I}$ for generative models on our unified encoder. As $\mathbf{D}$ and $\mathbf{I}$ increase, solution quality improves markedly in the early stages and then gradually saturates. Overall, the performance-efficiency trade-off remains favorable, since most of the gap reduction is achieved with only a small number of steps.

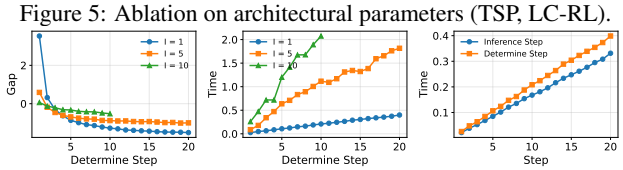


Figure 6: Effect of determine/inference steps on MCut for AE.

RQ3: Effect of Augmentation Samples with the LC Paradigm. This augmentation follows MatNet and is rooted in the randomness of initial node embeddings. As in Fig. 7, under unified UECO, as

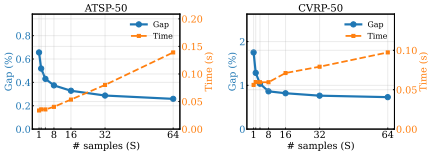


Figure 7: Effect of aug. samples for LC.

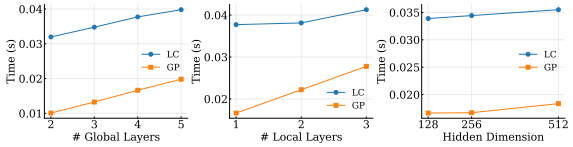


Figure 8: Solving efficiency vs. parameter quantity.

S increases, the gap sharply decreases while the inference time grows approximately linearly. The performance gain exhibits diminishing returns beyond a moderate number of samples.

RQ4: Memory and Runtime Efficiency Study. Theoretically, for each global block, the L_{local}-step edge-evolving module requires approximately $O(L_{\text{local}}|\mathcal{E}|d^2)$ computation, while the global attention and score-mixture operations require $O(|\mathcal{V}|^2 d + |\mathcal{V}|d^2)$. The principal activation memory is $O(|\mathcal{E}|d + |\mathcal{V}|^2 h)$. Fig. 8 reveals how the inference time scales with increased model parameters under UECO. We further profile the computational overhead in Appendix E.7. The log-log scaling exponent for UECO's inference time is ~ 1.12, and ~ 1.15 for memory, supporting tractable complexity and acceptable performance-efficiency trade-off in practice.

RQ5: Applicability Beyond Graph-based Tasks. 1) Knapsack Problem (KP). We follow the KP-50 setting in [11]. Each instance contains 50 items with weights and values independently sampled from a uniform distribution $\mathcal{U}(0, 1)$, with the knapsack capacity fixed to 12.5. **2) Flexible Flow Shop Problem (FFSP).** We consider the official FFSP-20 setting from MatNet [13], consisting of 20 jobs, three processing stages, and four unrelated parallel machines per stage. Following the original protocol, UECO is trained from scratch on online-generated instances for 2000 epochs; evaluation uses the same test data and baselines. Together, KP and FFSP extend the evidence from routing and graph-selection tasks to constrained subset selection and multi-stage scheduling.

Table 13: Results on KP-50.

Method	Category	Obj.↑	Gap↓
OR-Tools	OR	20.021	0.000%
AM (G)	LC-RL(OS)-Trans.	19.986	0.173%
POMO (G)	LC-RL(OS)-Trans.	19.995	0.130%
UECO (G)	LC-RL(OS)-Uni.	**20.016**	**0.024%**

Table 14: Results on FFSP-20. The objective is the average makespan.

Method	Category	Obj.↓
CPLEX (600s)	OR	36.6
Shortest Job First	Heuristic	31.3
Genetic Algorithm	Heuristic	30.6
Particle Swarm Opt.	Heuristic	29.1
MatNet (G)	LC-RL(OS)-Trans.	27.3
UECO (G)	LC-RL(OS)-Uni.	**27.0**

RQ6: Deeper Hierarchical Ablations. Beyond (already) showing ***whether UECO works***, Appendix E.8 pinpoints mechanistically ***what matters and why***, from perturbations of edge evolution and mixed attention, to module-level removals (e.g., dual-channel design, local-global blocks).

RQ7: Statistical Significance. In Appendix E.9, we compare the solved objectives with the optimal ones via box-plots, indicating statistical stability of solutions delivered by UECO across benchmarks.

5.4 Key Empirical Takeaways

1) Broad effectiveness under controlled substitution. Replacing only the host encoder with UECO yields broad and near-consistent gains across seven COPs and LC, GP, and AE pipelines while preserving their original objectives, decoders, and feasibility mechanisms. **2) The gains arise from complementary local-global modeling.** Ablations attribute the gains to recurrent edge evolution and structure-modulated global attention rather than parameter count alone. The dual-channel design is particularly important when directional pairwise structure matters, as in ATSP/ACVRP. **3) A stronger encoder can extend paradigm capability.** UECO enables LC-RL to reach a 1.632% gap on TSP-500 without 2-OPT, extrapolate to TSP-1K, and transfer to TSP-100K. This suggests that some apparent LC-RL scalability limits are representational rather than intrinsic. **4) Additional computation offers diminishing returns.** Greater capacity particularly benefits GP, while additional inference steps and LC augmentation provide diminishing gains. Together with the different effects of 2-OPT (discussed in Appendix F), these results reveal a tunable quality-efficiency trade-off.

6 Conclusion

We propose UECO, a shared encoder that reduces dependence on paradigm-specific backbone choices in NCO. By iteratively evolving edge representations and using them to modulate global attention, UECO combines complementary local structural modeling and global contextual reasoning within a depth-wise closed loop. We offer UECO as a reusable backbone reference for cumulative development of NCO, with both performance improved and higher-level paradigms and training designs examined in a more consistent architectural space. We provide further discussions on empirical observations, limitations and future work, and broader impact in Appendices F, G, and H.

Acknowledgements

This work was partly supported by NSFC (92370201) and Shanghai Municipal Universities Industry-Academia-Research Joint Innovation Program.

References

[1] G. Singh and M. Rizwanullah, "Combinatorial optimization of supply chain networks: A retrospective & literature review," *Materials today: proceedings*, vol. 62, pp. 1636–1642, 2022.

[2] L. Baty, K. Jungel, P. S. Klein, A. Parmentier, and M. Schiffer, "Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows," *Transportation Science*, 2024.

[3] J. Yan, J. Liu, Y. Tu, F. Zhou, W. Pan, Z. Gui, and L. Shi, "Design linear constrained neural layers with implicit convex optimization," in *Forty-third International Conference on Machine Learning*, 2026.

[4] R. Wang, Z. Guo, W. Pan, J. Ma, Y. Zhang, N. Yang, Q. Liu, L. Wei, H. Zhang, C. Liu, Z. Jiang, X. Yang, and J. Yan, "Pygmtools: A python graph matching toolkit," *Journal of Machine Learning Research*, vol. 25, no. 33, pp. 1–7, 2024.

[5] Y. Li, J. Ma, Y. Yang, Q. Wu, H. Zha, and J. Yan, "Generative modeling reinvents supervised learning: Label repurposing with predictive consistency learning." in *ICML*, 2025.

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

[7] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[8] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[9] F. Berto, C. Hua, J. Park, M. Kim, H. Kim, J. Son, H. Kim, J. Kim, and J. Park, "RL4CO: a unified reinforcement learning for combinatorial optimization library," in *Neural Information Processing Systems 2023, Workshop: New Frontiers in Graph Learning*, 2023.

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

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

[12] M. Kim, J. Park, and J. Park, "Sym-nco: Leveraging symmetricity for neural combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1936–1949, 2022.

[13] Y.-D. Kwon, J. Choo, I. Yoon, M. Park, D. Park, and Y. Gwon, "Matrix encoding networks for neural combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5138–5149, 2021.

[14] W. Pan, N. Chen, J. Liu, J. Ma, and J. Yan, "Corectifier: Hierarchically rectifying policy rollouts for reinforcement learning in combinatorial solvers," in *The Fortieth Annual Conference on Neural Information Processing Systems*, 2026.

[15] C. K. Joshi, T. Laurent, and X. Bresson, "An efficient graph convolutional network technique for the travelling salesman problem," *arXiv preprint arXiv:1906.01227*, 2019.

[16] Z.-H. Fu, K.-B. Qiu, and H. Zha, "Generalize a small pre-trained model to arbitrarily large tsp instances," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 7474–7482.

[17] Y. Li, J. Guo, R. Wang, and J. Yan, "From distribution learning in training to gradient search in testing for combinatorial optimization," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[19] S. Ahn, Y. Seo, and J. Shin, "Learning what to defer for maximum independent sets," in *International Conference on Machine Learning*. PMLR, 2020.

[20] S. Zhou, Y. Ding, C. Zhang, Z. Cao, and Y. Jin, "Dualopt: A dual divide-and-optimize algorithm for the large-scale traveling salesman problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, February 2025.

[21] J. Ma, W. Pan, Y. Li, and J. Yan, "COExpander: Adaptive solution expansion for combinatorial optimization," in *Forty-second International Conference on Machine Learning*, 2025.

[22] Y. Wang, Y. Li, J. Ma, J. Yan, and Y. Chang, "Nexco: Native solution expansion for diffusion-based combinatorial optimization," in *The Fourteenth International Conference on Learning Representations*, 2026.

[23] Y. Li, L. Chen, H. Wang, R. Wang, and J. Yan, "Generation as search operator for test-time scaling of diffusion-based combinatorial optimization," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[24] W. Pan, H. Xiong, J. Ma, W. Zhao, Y. Li, and J. Yan, "UniCO: On unified combinatorial optimization via problem reduction to matrix-encoded general TSP," in *The Thirteenth International Conference on Learning Representations*, 2025.

[25] X. Bresson and T. Laurent, "Residual gated graph convnets," *arXiv preprint arXiv:1711.07553*, 2018.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[27] A. Lischka, J. Wu, M. H. Chehreghani, and B. Kulcsár, "A GREAT architecture for edge-based graph problems like tsp," *arXiv preprint arxiv:2408.16717*, 2024.

[28] D. Meng, Z. Cao, Y. Wu, Y. Hou, H. Ge, and Q. Zhang, "EFormer: An effective edge-based transformer for vehicle routing problems," in *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, J. Kwok, Ed. International Joint Conferences on Artificial Intelligence Organization, 2025, pp. 8582–8590.

[29] D. Meng, Z. Cao, J. Gao, Y. Wu, and Y. Hou, "UniteFormer: Unifying node and edge modalities in transformers for vehicle routing problems," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[30] M. Kim, J. Park, and J. Kim, "Learning collaborative policies to solve np-hard routing problems," in *Advances in neural information processing systems*, 2021.

[31] D. Drakulic, S. Michel, F. Mai, A. Sors, and J.-M. Andreoli, "BQ-NCO: Bisimulation quotienting for efficient neural combinatorial optimization," in *Advances in Neural Information Processing Systems*, 2023.

[32] D. Drakulic, S. Michel, and J.-M. Andreoli, "GOAL: A generalist combinatorial optimization agent learner," in *The Thirteenth International Conference on Learning Representations*, 2025.

[33] R. Qiu, Z. Sun, and Y. Yang, "DIMES: A differentiable meta solver for combinatorial optimization problems," in *Advances in Neural Information Processing Systems 35*, 2022.

[34] S. Sanokowski, W. Berghammer, S. Hochreiter, and S. Lehner, "Variational annealing on graphs for combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 36, pp. 63 907–63 930, 2023.

[35] H. P. Wang and P. Li, "Unsupervised learning for combinatorial optimization needs meta learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[36] S. Sanokowski, S. Hochreiter, and S. Lehner, "A diffusion model framework for unsupervised neural combinatorial optimization," in *Forty-first International Conference on Machine Learning*, 2024.

[37] Y. Li, J. Guo, R. Wang, H. Zha, and J. Yan, "Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[38] Y. Wang, Y. Li, J. Yan, and Y. Chang, "Strudico: Structured denoising diffusion with gradient-free inference-stage boosting for memory and time efficient combinatorial optimization," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[39] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," *Advances in neural information processing systems*, vol. 32, 2019.

[40] Y. Ma, Z. Cao, and Y. M. Chee, "Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[41] J. Ma, W. Pan, B. Cai, X. Zhang, and J. Yan, "NLD4CO: Neural langevin dynamics for combinatorial optimization," in *The Fortieth Annual Conference on Neural Information Processing Systems*, 2026.

[42] J. Zheng, K. He, J. Zhou, Y. Jin, and C.-M. Li, "Combining reinforcement learning with lin-kernighan-helsgaun algorithm for the traveling salesman problem," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, pp. 12 445–12 452, May 2021.

[43] L. Xin, W. Song, Z. Cao, and J. Zhang, "Neurolkh: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[44] W. Kool, H. van Hoof, J. Gromicho, and M. Welling, "Deep policy dynamic programming for vehicle routing problems," in *International conference on integration of constraint programming, artificial intelligence, and operations research*. Springer, 2022, pp. 190–213.

[45] H. Ye, J. Wang, Z. Cao, H. Liang, and Y. Li, "Deepaco: neural-enhanced ant systems for combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[46] H. Xiong, W. Pan, J. Huang, Z. Xia, and J. Yan, "From graph embedding to LKH: Bridging learning and heuristics for a streamlined general TSP solver," 2024.

[47] M. Kim, S. Choi, H. Kim, J. Son, J. Park, and Y. Bengio, "Ant colony sampling with GFlownets for combinatorial optimization," in *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

[48] S. Li, Z. Yan, and C. Wu, "Learning to delegate for large-scale vehicle routing," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 198–26 211, 2021.

[49] H. Ye, J. Wang, H. Liang, Z. Cao, Y. Li, and F. Li, "GLOP: Learning global partition and local construction for solving large-scale routing problems in real-time," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[50] Q. Hou, J. Yang, Y. Su, X. Wang, and Y. Deng, "Generalize learned heuristics to solve large-scale vehicle routing problems in real-time," in *The Eleventh International Conference on Learning Representations*, 2023.

[51] Z. Zheng, C. Zhou, T. Xialiang, M. Yuan, and Z. Wang, "UDC: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[52] F. Luo, X. Lin, F. Liu, Q. Zhang, and Z. Wang, "Neural combinatorial optimization with heavy decoder: Toward large scale generalization," in *The 37th Anniversary Conference on Neural Information Processing Systems*, 2023.

[53] J. He, C. Yang, and J. Ji, "GDaT: Generalizable density-aware transformer for solving the traveling salesman problem," 2026.

[54] C. Zhou, X. Lin, Z. Wang, X. Tong, M. Yuan, and Q. Zhang, "Instance-conditioned adaptation for large-scale generalization of neural combinatorial optimization," *CoRR*, 2024.

[55] Y. Li, J. Ma, W. Pan, R. Wang, H. Geng, N. Yang, and J. Yan, "Unify ml4tsp: Drawing methodological principles for tsp and beyond from streamlined design space of learning and search," in *The Thirteenth International Conference on Learning Representations*, 2025.

[56] H. Yuan, W. Ouyang, C. Zhang, C. Li, and Y. Sun, "OPTFM: A scalable multi-view graph transformer for hierarchical pre-training in combinatorial optimization," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[57] J. Ma, W. Pan, Y. Li, and J. Yan, "Ml4co-bench-101: Benchmark machine learning for classic combinatorial problems on graphs," in *Advances in Neural Information Processing Systems*, 2025.

[58] H. Yi, Z. Huang, Z. Cao, and Y. Ma, "RADAR: Learning to route with asymmetry-aware distance representations," in *International Conference on Learning Representations (ICLR)*, 2026.

[59] K. Xu, F. Boussemart, F. Hemery, and C. Lecoutre, "A simple model to generate hard satisfiable instances," *arXiv preprint cs/0509032*, 2005.

[60] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hungar. Acad. Sci*, vol. 5, pp. 17–61, 1960.

[61] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[62] L. Jure, "Snap datasets: Stanford large network dataset collection," *Retrieved December 2021 from http://snap. stanford. edu/data*, 2014.

[63] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.

[64] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, "Concorde tsp solver," 2006.

[65] K. Helsgaun, "An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems," *Roskilde: Roskilde University*, vol. 12, 2017.

[66] S. Lamm, P. Sanders, C. Schulz, D. Strash, and R. F. Werneck, "Finding near-optimal independent sets at scale," in *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)*. SIAM, 2016, pp. 138–150.

[67] L. Gurobi Optimization, "Gurobi optimizer reference manual (2020)," 2023.

[68] Y. Min, Y. Bai, and C. P. Gomes, "Unsupervised learning for solving the travelling salesman problem," in *Advances in Neural Information Processing Systems*, 2023.

[69] B. Hudson, Q. Li, M. Malencia, and A. Prorok, "Graph neural network guided local search for the traveling salesperson problem," in *International Conference on Learning Representations*, 2022.

[70] M. Pan, G. Lin, Y.-W. Luo, B. Zhu, Z. Dai, L. Sun, and C. Yuan, "Preference optimization for combinatorial optimization problems," in *Forty-second International Conference on Machine Learning*, 2025.

[71] F. Luo, X. Lin, M. Zhong, F. Liu, Z. Wang, J. Sun, and Q. Zhang, "Learning to insert for constructive neural vehicle routing solver," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[72] H. Fang, Z. Song, P. Weng, and Y. Ban, "INViT: A generalizable routing problem solver with invariant nested view transformer," in *Forty-first International Conference on Machine Learning*, 2024.

[73] C. Gao, H. Shang, K. Xue, D. Li, and C. Qian, "Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*, 2024.

[74] J. Zhou, Y. Wu, W. Song, Z. Cao, and J. Zhang, "Towards omni-generalizable neural methods for vehicle routing problems," in *International Conference on Machine Learning*, 2023.

[75] P. R. d. O. da Costa, J. Rhuggenaath, Y. Zhang, and A. Akcay, "Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning," in *Proceedings of The 12th Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. J. Pan and M. Sugiyama, Eds., vol. 129. PMLR, 18–20 Nov 2020, pp. 465–480.

[76] Z. Sun and Y. Yang, "DIFUSCO: Graph-based diffusion solvers for combinatorial optimization," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[77] H. Sun, K. Goshvadi, A. Nova, D. Schuurmans, and H. Dai, "Revisiting sampling for combinatorial optimization," in *International Conference on Machine Learning*. PMLR, 2023, pp. 32 859–32 874.

[78] S. Feng and Y. Yang, "Regularized langevin dynamics for combinatorial optimization," in *Forty-second International Conference on Machine Learning*, 2025.

[79] S. Wang, Z. Guo, C. Lu, and J. Yan, "Fractional langevin dynamics for combinatorial optimization via polynomial-time escape," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[80] W. Pan, J. Ma, N. Chen, Y. Li, and J. Yan, "Problem distributions as tasks: Repurposing meta learning for generative combinatorial optimization towards multi-task pretraining and adaptation," in *Forty-third International Conference on Machine Learning*, 2026.

[81] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," *Physical review E*, vol. 65, no. 2, p. 026107, 2002.

[82] Z. Huang, J. Zhou, Z. Cao, and Y. XU, "Rethinking light decoder-based solvers for vehicle routing problems," in *The Thirteenth International Conference on Learning Representations*, 2025.

[83] F. Berto, C. Hua, J. Park, L. Luttmann, Y. Ma, F. Bu, J. Wang, H. Ye, M. Kim, S. Choi, N. G. Zepeda, A. Hottung, J. Zhou, J. Bi, Y. Hu, F. Liu, H. Kim, J. Son, H. Kim, D. Angioni, W. Kool, Z. Cao, J. Zhang, K. Shin, C. Wu, S. Ahn, G. Song, C. Kwon, L. Xie, and J. Park, "RL4CO: an Extensive Reinforcement Learning for Combinatorial Optimization Benchmark," *arXiv preprint arXiv:2306.17100*, 2024.

[84] C. Wang and T. Yu, "Efficient training of multi-task combinarotial neural solver with multi-armed bandits," *arXiv preprint arXiv:2305.06361*, 2023.

[85] J. Zhou, Z. Cao, Y. Wu, W. Song, Y. Ma, J. Zhang, and C. Xu, "Mvmoe: Multi-task vehicle routing solver with mixture-of-experts," *arXiv preprint arXiv:2405.01029*, 2024.

[86] F. Liu, X. Lin, Z. Wang, Q. Zhang, X. Tong, and M. Yuan, "Multi-task learning for routing problem with cross-problem zero-shot generalization," *arXiv preprint arXiv:2402.16891*, 2024.

[87] X. Jiang, Y. Wu, Y. Wang, and Y. Zhang, "UNCO: Towards unifying neural combinatorial optimization through large language model," *arXiv preprint arXiv:2408.12214*, 2024.

[88] Z. Liao, J. Chen, D. Wang, Z. Zhang, and J. Wang, "BOPO: Neural combinatorial optimization via best-anchored and objective-guided preference optimization," in *Forty-second International Conference on Machine Learning*, 2025.

[89] Z. Fang, D. Wang, J. Chen, J. Wang, and Z. Zhang, "UCPO: A universal constrained combinatorial optimization method via preference optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[90] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2018.

Appendix

Contents

A Supplementary Introduction to the Backgrounds for NCO **18**

A.1 Mainstream Decision Paradigms 18

A.1.1 Global Prediction (GP) 18

A.1.2 Local Construction (LC) 18

A.1.3 Adaptive Expansion (AE) 18

A.2 Mainstream Model Backbones 19

A.2.1 Node-centric (Graph) Transformers 19

A.2.2 Anisotropic Graph Convolutional Networks (GCNs) 20

A.2.3 Edge-based (Graph) Transformers 21

A.3 Mainstream Learning Objectives 22

A.3.1 Supervised Solution Proximity 22

A.3.2 Solution Distribution Proximity 22

A.3.3 Reinforcement Solving Objective 22

B Formal Problem Definitions **22**

B.1 Edge-oriented Tasks 22

B.2 Node-oriented Tasks 23

B.3 Non-graph-based Tasks 23

C Additional Clarification on Our Positioning **23**

C.1 Evaluation Scope Comparison within Related Works 23

C.2 Further Comparison against Edge-aware Attention Works for NCO 25

D Detailed Settings for Training and Solving **25**

D.1 Hardware Configuration 25

D.2 Hyper-parameters and Model Settings 26

E Supplementary Experimental Results **27**

E.1 Full Results on TSP with 2OPT Enabled for Several Baselines 27

E.2 Additional Baseline Comparisons (INViT and L2C) 27

E.3 Additional Results on ACVRP 28

E.4 Additional Results on Cross-distribution Generalization (6 settings) 28

E.5 Additional Results on UECO with D&C Pipelines 28

E.6 Per-instance Results on TSPLIB and CVRPLIB 29

E.7 Memory and Runtime Overhead 29

E.8 In-depth Hierarchical Ablations: Whether It Works, What Matters, and Why? . . . 30

E.9 Results for Statistical Stability 32

F Additional Observations and Discussions **34**

F.1 Rethinking LC4CO: Towards Competitive Scalability upon UECO 34

F.2 Diverged Efficacy of Post-inference Improving Techniques across Paradigms . . . 34

G Limitation and Future Work **35**

H Broader Impact **35**

A Supplementary Introduction to the Backgrounds for NCO

A.1 Mainstream Decision Paradigms

According to [21, 57], the process of learning how to construct a solution for COP can be viewed as Variable Determination Process (VDP). This process gradually transitions from a state where all decision variables are completely undetermined to a fully determined state. Based on the relationship between the number of decision variables N and the number of steps k, VDP can be categorized into three major decision paradigms below.

A.1.1 Global Prediction (GP)

The Global Prediction paradigm (corresponding to $k = 1$) is an one-shot decoding process, where the neural network processes the problem instance $\mathcal{I} \sim p(\mathcal{G})$ and predicts a global probability heatmap $\hat{\mathbf{H}}$ (equivalently, a two-channel logit map $\mathbf{Z}_{\text{GP}}$ in this paper) over all decision variables simultaneously, representing the likelihood of each node or edge belonging to the optimal set [15, 76]. The final discrete solution is subsequently obtained via non-learnable decoding heuristics, such as greedy search or beam search [37]. While GP approaches benefit from a holistic view of the graph structure and high inference efficiency, they often suffer from over-smoothing in large-scale instances [21], where indistinguishable probability values hinder effective decoding. Furthermore, since variables are predicted independently or essentially in parallel, GP methods may struggle to resolve complex constraint conflicts during the post-hoc decoding phase.

A.1.2 Local Construction (LC)

The Local Construction paradigm (corresponding to $k = N$) formulates solving COPs as a Markov Decision Process (MDP) or a sequential generation task. LC solvers [11, 10] construct the solution auto-regressively by selecting exactly one decision variable at each step conditioned on the current partial state. This step-by-step approach ensures strictly feasible solutions by masking invalid actions and effectively leveraging local state information [83]. However, the reliance on myopic, "next-token" prediction often results in a lack of global perspective. Additionally, the computational cost of LC solvers scales unfavorably with problem size due to the onerous nature of sequential auto-regression.

A.1.3 Adaptive Expansion (AE)

Proposed by [21], Adaptive Expansion represents a hybrid paradigm ($1 < k < N$) designed to bridge the dichotomy between GP and LC. Rather than determining variables all at once or one by one, AE employs an iterative "partial-to-full" mechanism. In each round, a global predictor generates an informative heatmap conditioned not only on the graph features but also on the current partial solution (a set of already fixed variables). This heatmap guides a determination operator to expand the partial solution by fixing a subset of the most confident variables, allowing for adaptive step sizes. This paradigm effectively narrows the search space using partial guidance, mitigating the noise issues of GP while avoiding the computational inefficiency and myopia of strict auto-regression found in LC.

A.2 Mainstream Model Backbones

A.2.1 Node-centric (Graph) Transformers

AM [10] adapts the standard Transformer [26] architecture for learning to solve routing problems. The model follows an encoder-decoder structure where the encoder extracts structural features and the decoder autoregressively constructs the solution.

Encoder. The encoder transforms raw node features $\mathbf{x}_i$ into high-dimensional embeddings. The initial embedding $\mathbf{h}_i^{(0)}$ is derived via a linear projection with learnable weights $\mathbf{W}^x$ and bias $\mathbf{b}^x$:

$$\mathbf{h}_i^{(0)} = \mathbf{W}^x \mathbf{x}_i + \mathbf{b}^x. \tag{13}$$

The architecture consists of N layers (typically $N = 3$), where each layer ℓ contains a Multi-Head Attention (MHA) mechanism followed by a node-wise Feed-Forward (FF) network. Batch Normalization (BN) and residual connections are applied to stabilize training. The update rule for the ℓ-th layer is defined as:

$$\hat{\mathbf{h}}_i = \mathrm{BN}^\ell \left(\mathbf{h}_i^{(\ell-1)} + \mathrm{MHA}_i^\ell \left(\mathbf{h}_1^{(\ell-1)}, \ldots, \mathbf{h}_n^{(\ell-1)} \right) \right), \tag{14}$$

$$\mathbf{h}_i^{(\ell)} = \mathrm{BN}^\ell \left(\hat{\mathbf{h}}_i + \mathrm{FF}^\ell(\hat{\mathbf{h}}_i) \right). \tag{15}$$

The MHA module allows nodes to aggregate information from neighbors based on compatibility, while the FF module processes each node independently. Upon passing through all N layers, the model produces the final node embeddings $\mathbf{h}_i^{(N)}$. Additionally, a global graph embedding $\overline{\mathbf{h}}^{(N)}$ is computed by averaging the embeddings of all nodes to capture the holistic graph context:

$$\overline{\mathbf{h}}^{(N)} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{h}_i^{(N)}. \tag{16}$$

Decoder. The decoder generates the solution sequence π sequentially. To effectively query the next node to visit at step t, AM constructs a dynamic "context node" embedding $\mathbf{h}_{(c)}$. This vector aggregates the static global graph info with the dynamic state of the partial solution, specifically the graph embedding $\overline{\mathbf{h}}^{(N)}$, the embedding of the previous node π_{t-1}, and the first node π_1 (to facilitate the return to the depot in TSP):

$$\mathbf{h}_{(c)}^{(N)} = \begin{cases} \overline{\mathbf{h}}^{(N)} \,||\, \mathbf{h}_{\pi_{t-1}}^{(N)} \,||\, \mathbf{h}_{\pi_1}^{(N)} & t > 1 \\ \overline{\mathbf{h}}^{(N)} \,||\, \mathbf{v}^l \,||\, \mathbf{v}^f & t = 1 \end{cases}, \tag{17}$$

where $\mathbf{v}^l$ and $\mathbf{v}^f$ are learnable parameters for the initial step. This context vector serves as the query in a specific glimpse attention mechanism over the encoder outputs.

To compute the probability of selecting the next node, the context query $\mathbf{q}_{(c)}$ attends to the keys $\mathbf{k}_j$ (and corresponding values) derived from the node embeddings. A crucial design in AM is the use of a $\tanh$-clipped compatibility score to regulate the logits and improve exploration:

$$u_{(c)j} = \begin{cases} C \cdot \tanh\left(\frac{\mathbf{q}_{(c)}^\top \mathbf{k}_j}{\sqrt{d_h}} \right) & j \notin \{\pi_1, \ldots, \pi_{t-1}\} \\ -\infty & \text{otherwise} \end{cases}, \tag{18}$$

where C is a constant (e.g., 10), and the masking ensures validity by preventing the selection of previously visited nodes. Finally, the selection probabilities are obtained via a softmax distribution:

$$p_\theta(\pi_t = i \mid \mathcal{I}, \pi_{1:t-1}) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}}. \tag{19}$$

POMO [11] builds upon the foundational encoder-decoder architecture of AM [10] but significantly modifies the training and decoding paradigms to leverage problem symmetries. By introducing a "multiple optima" approach, it generates diverse solutions in parallel to enhance both baseline estimation and solution space exploration.

Unlike AM, which relies on a single pair of learnable parameters to initialize decoding, POMO generates N parallel trajectories for a problem instance of size N, with each node serving as a distinct starting point. This strategy enforces a symmetric exploration of the solution space. Consequently, for the i-th trajectory initiated at node a_1^i, the decoder state is initialized directly with the encoder's output embedding for that node:

$$\mathbf{h}_{\pi_1^i} = \mathbf{h}_{a_1^i}. \tag{20}$$

This design effectively removes the need for the trainable start token placeholders (v^l, v^f) required in the original AM formulation.

For subsequent time steps ($t > 1$), the decoding process mirrors the structure of AM but is adapted for parallel execution. The context embedding $\mathbf{h}_{(c)}^i$ for the i-th trajectory is constructed by fusing three distinct features: the global graph embedding $\overline{\mathbf{h}}^{(N)}$, the embedding of the current node π_{t-1}^i, and the embedding of the trajectory-specific starting node π_1^i:

$$\mathbf{h}_{(c)}^i = \overline{\mathbf{h}}^{(N)} \,||\, \mathbf{h}_{\pi_{t-1}^i} \,||\, \mathbf{h}_{\pi_1^i}. \tag{21}$$

To maintain computational efficiency, the context embeddings for all N trajectories are stacked into a single batch. This allows the attention mechanism to process the decoding steps for all symmetric views simultaneously.

Since its proposal, this symmetric parallel decoding scheme has established a de facto standard in the community, guiding the architectural design of numerous subsequent solvers within the LC paradigm.

A.2.2 Anisotropic Graph Convolutional Networks (GCNs)

GCN4CO [15] takes the lead to apply the GCN [18] to NCO solving for its capability of capturing topological information in locality. Specifically, they propose a deep Graph Convolutional Network that operates on the graph representation of the problem, where nodes are initialized with coordinate features $\mathbf{x}_i$ and edges with distance features $\mathbf{e}_{ij}$. The core component is the anisotropic message-passing mechanism, where edge features dynamically modulate the information flow between neighboring nodes.

Formally, let $\mathbf{h}_i^\ell$ and $\mathbf{e}_{ij}^\ell$ denote the node and edge embeddings at layer ℓ, respectively. The update rule for the edge embeddings integrates information from the connected nodes:

$$\mathbf{e}_{ij}^{\ell+1} = \mathbf{e}_{ij}^\ell + \text{ReLU}\left(\text{BN}\left(\mathbf{W}_3^\ell \mathbf{e}_{ij}^\ell + \mathbf{W}_4^\ell \mathbf{h}_i^\ell + \mathbf{W}_5^\ell \mathbf{h}_j^\ell\right)\right), \tag{22}$$

where BN denotes Batch Normalization and $\mathbf{W}$ are learnable weights. Crucially, the node embeddings are updated via a gated aggregation scheme, where the influence of neighbor j on node i is weighted by a gate η_{ij}^ℓ derived from the edge features:

$$\mathbf{h}_i^{\ell+1} = \mathbf{h}_i^\ell + \text{ReLU}\left(\text{BN}\Big(\mathbf{W}_1^\ell \mathbf{h}_i^\ell + \sum_{j \in \mathcal{N}(i)} \eta_{ij}^\ell \odot \mathbf{W}_2^\ell \mathbf{h}_j^\ell\Big)\right), \quad \text{with} \quad \eta_{ij}^\ell = \frac{\sigma(\mathbf{e}_{ij}^\ell)}{\sum_{j' \in \mathcal{N}(i)} \sigma(\mathbf{e}_{ij'}^\ell) + \epsilon}, \tag{23}$$

where σ is the sigmoid function and ϵ is a stability constant. This gating mechanism renders the convolution "anisotropic," allowing the model to focus on promising connections. Finally, the edge embeddings from the last layer are passed through an MLP classifier to produce a probabilistic heatmap $\hat{\mathbf{H}} \in [0,1]^{|\mathcal{V}| \times |\mathcal{V}|}$, representing the likelihood of each edge belonging to the optimal solution.

This has fostered numerous works of the GP (as well as AE) paradigm, enabling effective production of probability heatmaps over either node- or edge-decision tasks, and works seamlessly with the emergence of discrete diffusion scheme.

Specifically, **DIFUSCO** [76] formulates combinatorial optimization as a discrete denoising process. It treats the solution as a binary vector $\mathbf{s} \in \{0,1\}^N$ and corrupts it via a forward diffusion process. The solver effectively learns to reverse this process by iteratively reconstructing the clean solution. At each step t, the model predicts a denoised estimate $\tilde{\mathbf{s}}_0$ to guide the reverse transition $p_\theta(\mathbf{s}_{t-1}|\mathbf{s}_t)$:

$$p_\theta(\mathbf{s}_{t-1}|\mathbf{s}_t) = \sum_{\tilde{\mathbf{s}}_0} q(\mathbf{s}_{t-1}|\mathbf{s}_t, \tilde{\mathbf{s}}_0) p_\theta(\tilde{\mathbf{s}}_0|\mathbf{s}_t), \tag{24}$$

where $q(\mathbf{s}_{t-1}|\mathbf{s}_t, \tilde{\mathbf{s}}_0)$ is the tractable posterior of the forward diffusion. While powerful, this iterative paradigm suffers from high latency due to the requirement of hundreds of neural function evaluations (NFEs) to traverse from noise $\mathbf{s}_T$ to solution $\mathbf{s}_0$.

Addressing this bottleneck, **Fast-T2T** [37] introduces the concept of *Optimization Consistency* to accelerate the solving process. Instead of stepping through the entire reverse chain, it trains a consistency model $f_\theta(\mathbf{s}_t, t)$ to map any arbitrary noisy state $\mathbf{s}_t$ directly to the optimal solution $\mathbf{s}^*$. This is achieved by minimizing the consistency loss that enforces predictions from different time steps to align with the same ground truth:

$$\mathcal{L}_{\text{OptCM}}(\theta) = \mathbb{E}\left[d(f_\theta(\mathbf{s}_t, t), \delta(\mathbf{s} - \mathbf{s}^*)) + d(f_\theta(\mathbf{s}_{t'}, t'), \delta(\mathbf{s} - \mathbf{s}^*))\right]. \tag{25}$$

By learning this direct mapping, Fast-T2T enables one-step generation of the final probability heatmap $p_\theta(\mathbf{s}_0|\mathbf{s}_t) \approx f_\theta(\mathbf{s}_t, t)$, drastically reducing inference time while maintaining high solution quality.

A.2.3 Edge-based (Graph) Transformers

Proposed to address the limitations of node-centric models in capturing complex topological structures, **GREAT** (Graph Edge Attention Network) [27] introduces a purely edge-focused architecture. Unlike standard GNNs that treat edges as auxiliary features, GREAT elevates edges to primary citizens, learning representations directly in the edge space to better handle non-Euclidean routing problems (e.g., asymmetric TSP).

The core mechanism is the *Edge-Integrated Attention*, which updates the embedding $\mathbf{e}_{ij}$ of an edge (i, j) by aggregating information from its incident edges. Specifically, the attention coefficients α_{ijk} measure the importance of a neighboring edge (j, k) to the current edge (i, j):

$$\alpha_{ijk} = \text{softmax}_k\left(\frac{(\mathbf{W}^Q\mathbf{e}_{ij})^\top(\mathbf{W}^K\mathbf{e}_{jk})}{\sqrt{d_k}}\right), \tag{26}$$

where $\mathbf{W}^Q, \mathbf{W}^K$ are learnable projection matrices. The edge embedding is then updated by aggregating the value vectors of neighbors, fused with a residual connection:

$$\mathbf{e}_{ij}^{(\ell+1)} = \text{LN}\left(\mathbf{e}_{ij}^{(\ell)} + \sum_{k\in\mathcal{N}(j)} \alpha_{ijk}(\mathbf{W}^V\mathbf{e}_{jk}^{(\ell)})\right), \tag{27}$$

where LN denotes Layer Normalization. This design allows GREAT to propagate compositional messages along paths (i.e., edge-to-edge), making it particularly effective for tasks requiring precise link prediction and sparsification.

EFormer (Edge-augmented Transformer) [28] pushes the edge-centric paradigm further by using the edge features (e.g., distance matrix) as the *sole* input, eliminating the dependency on explicit node coordinates. It employs a *Precoder* to initialize node embeddings from edges and a *Parallel Encoding Strategy* to process topological and sequence information simultaneously.

First, the Precoder generates temporary node embeddings $\mathbf{h}_i^{(0)}$ via a mixed-score attention mechanism that aggregates incident edge features $\mathbf{e}_{ij}$:

$$\mathbf{h}_i^{(0)} = \sum_{j\in\mathcal{N}(i)} \text{softmax}_j(\mathbf{e}_{ij}\mathbf{W}^S) \odot (\mathbf{e}_{ij}\mathbf{W}^P), \tag{28}$$

where $\mathbf{W}^S$ and $\mathbf{W}^P$ project edges into scores and prototypes, respectively. Subsequently, EFormer utilizes a dual-stream encoder. The *Graph Stream* updates edge embeddings $\mathbf{E}^{(\ell)}$ using Residual Gated GCNs to capture global topology, while the *Node Stream* updates node embeddings $\mathbf{H}^{(\ell)}$ via standard Self-Attention to capture long-range dependencies. The two streams interact to enrich the representation:

$$\mathbf{E}^{(\ell+1)} = \text{GatedGCN}(\mathbf{E}^{(\ell)}, \mathbf{H}^{(\ell)}), \quad \mathbf{H}^{(\ell+1)} = \text{MHA}(\mathbf{H}^{(\ell)}). \tag{29}$$

Finally, a decoding scheme with *Multiple-Query Integration* fuses information from both streams to autoregressively construct the solution, ensuring that decision-making is grounded in both global graph structure and local node states.

A.3 Mainstream Learning Objectives

Following the taxonomy systematically proposed in [55], there are distinct categories of objectives based on the availability of supervision signals: supervised methods that leverage high-quality reference solutions to guide the learning, and unsupervised methods (predominantly reinforcement learning) that directly optimize the problem-specific objective function.

A.3.1 Supervised Solution Proximity

This objective treats the NCO task as a classification problem, encouraging the model's predicted heatmap to approximate the ground-truth solution. Concretely, for edge-decision tasks, the encoder produces a two-channel logit map $\mathbf{Z}_{\text{GP}} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times 2}$, which is converted into a probability heatmap $\hat{\mathbf{H}} \in [0,1]^{|\mathcal{V}|\times|\mathcal{V}|}$ (e.g., by taking the positive-class probability after softmax over the 2 channels). For node-decision tasks, $\mathbf{Z}_{\text{GP}} \in \mathbb{R}^{|\mathcal{V}|\times 2}$ similarly yields $\hat{\mathbf{H}} \in [0,1]^{|\mathcal{V}|}$. Let $\mathbf{S}^*$ denote the corresponding optimal indicator (reshaped from $\mathbf{s}^*$ when needed). To address the severe class imbalance in sparse labels, a weighted binary cross-entropy (BCE) loss is widely adopted:

$$\mathcal{L}_{\text{SL}} = -\sum_{i,j}\left[w_1 \cdot \mathbf{S}^*_{i,j} \cdot \log(\hat{\mathbf{H}}_{i,j}) + w_0 \cdot (1-\mathbf{S}^*_{i,j}) \cdot \log(1-\hat{\mathbf{H}}_{i,j})\right], \tag{30}$$

where w_1 and w_0 are weights assigned to positive and negative classes, respectively. This objective provides strong guidance but relies heavily on the availability of expensive optimal labels.

A.3.2 Solution Distribution Proximity

Deviating from single-point prediction, generative models (e.g., Diffusion, VAE) aim to learn the conditional probability distribution $p_\theta(\mathbf{s} \mid \mathcal{I})$ of high-quality solutions. The training objective is to maximize the likelihood of the reference data. In practice, this is typically optimized via the Evidence Lower Bound (ELBO), involving a reconstruction term and a regularization term on the latent variable Z:

$$\mathcal{L}_{\text{Gen}} = -\mathbb{E}_{q(Z|\mathbf{s},\mathcal{I})}\left[\log \frac{p_\theta(\mathbf{s}, Z \mid \mathcal{I})}{q(Z \mid \mathbf{s}, \mathcal{I})}\right] \geq \mathbb{E}\left[-\log p_\theta(\mathbf{s} \mid \mathcal{I})\right]. \tag{31}$$

By modeling the full distribution, this paradigm enhances the diversity of generated solutions, which is beneficial for downstream search strategies.

A.3.3 Reinforcement Solving Objective

In the absence of ground truth, Reinforcement Learning (RL) methods optimize the expected quality of the solution directly. The model typically parameterizes a stochastic policy $p_\theta(\mathbf{s} \mid \mathcal{I})$. The objective is to minimize the expected cost $\mathcal{J}(\theta) = \mathbb{E}_{p_\theta(\mathbf{s}|\mathcal{I})}[l(\mathbf{s}, \mathcal{I})]$. Gradients are commonly estimated using the REINFORCE algorithm with a baseline $b(\mathcal{I})$ to reduce variance:

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{p_\theta(\mathbf{s}|\mathcal{I})}\left[(l(\mathbf{s}, \mathcal{I}) - b(\mathcal{I}))\nabla \log p_\theta(\mathbf{s} \mid \mathcal{I})\right]. \tag{32}$$

This approach allows the model to explore the solution space autonomously, making it the dominant choice for constructive solvers (e.g., AM, POMO).

B Formal Problem Definitions

We evaluate our method on seven representative graph-based COPs and two additional non-graph-based problems. Let $\mathcal{I} \sim p(\mathcal{G})$ denote an instance with vertex set $\mathcal{V} = \{1, 2, \ldots, n\}$ and edge set $\mathcal{E}$. The problems are formalized as follows.

B.1 Edge-oriented Tasks

1) Traveling Salesman Problem (TSP). Given $\mathcal{I}$ and an $N \times N$ cost matrix $\mathbf{C}$ with $\mathbf{C}_{ij}$ denoting the cost of traversing edge $(i,j) \in \mathcal{E}$, TSP seeks a tour $\tau = (i_1, \ldots, i_n)$ minimizing $\sum_{k=1}^{N-1} \mathbf{C}_{i_k i_{k+1}} + \mathbf{C}_{i_N i_1}$.

2) Asymmetric Traveling Salesman Problem (ATSP). ATSP generalizes TSP by allowing asymmetric costs, i.e., $\mathbf{C}_{ij} = \mathbf{C}_{ji}$ need not hold for $i, j \in \mathcal{V}$. Following [13, 31, 49], we focus on the metric setting satisfying the triangle inequality: $\mathbf{C}_{ij} + \mathbf{C}_{jk} \geq \mathbf{C}_{ik}$ for any three distinct nodes i, j, k.

3) Capacitated Vehicle Routing Problem (CVRP). Given an instance $\mathcal{I}$ with a depot (node 1), a set of customers $\mathcal{V}' = \mathcal{V} \setminus \{1\}$ with demands d_i, a vehicle capacity Q, and a cost matrix $\mathbf{C}$, the goal is to find m routes $\{\tau_1, \ldots, \tau_m\}$ minimizing the total travel cost. Each route is a node sequence $\tau_k = (i_0^k, i_1^k, \ldots, i_{n_k}^k, i_{n_k+1}^k)$ with $i_0^k = i_{n_k+1}^k = 1$ and $i_r^k \in \mathcal{V}'$ for $r = 1, \ldots, n_k$, and its cost is $\sum_{r=0}^{n_k} \mathbf{C}_{i_r^k i_{r+1}^k}$. The routes must visit each customer exactly once and satisfy the capacity constraint $\sum_{r=1}^{n_k} d_{i_r^k} \leq Q$ for all k.

4) Asymmetric Capacitated Vehicle Routing Problem (ACVRP). Given an instance $\mathcal{I}$ with a depot (node 1), a set of customers $\mathcal{V}' = \mathcal{V} \setminus \{1\}$ with demands d_i, a vehicle capacity Q, and an asymmetric cost matrix $\mathbf{C} \in \mathbb{R}_{\geq 0}^{|\mathcal{V}| \times |\mathcal{V}|}$, the goal is to find m routes $\tau_1, \ldots, \tau_m$ minimizing the total directed travel cost. Each route is a node sequence $\tau_k = (i_0^k, i_1^k, \ldots, i_{n_k}^k, i_{n_k+1}^k)$ with $i_0^k = i_{n_k+1}^k = 1$ and $i_r^k \in \mathcal{V}'$ for $r = 1, \ldots, n_k$, and its cost is $\sum_{r=0}^{n_k} \mathbf{C}_{i_r^k i_{r+1}^k}$. Different from the symmetric CVRP, the travel cost is directional, i.e., generally $\mathbf{C}_{ij} \neq \mathbf{C}_{ji}$, while $\mathbf{C}_{ii} = 0$. The routes must visit each customer exactly once and satisfy the capacity constraint $\sum_{r=1}^{n_k} d_{i_r^k} \leq Q$ for all k. In our experiments, $\mathbf{C}$ is generated from random integer directed costs and then transformed into a directed metric closure so that the triangle inequality holds, followed by a scaling factor; customer demands are sampled as random integers and normalized according to the problem size.

B.2 Node-oriented Tasks

5) Maximum Independent Set (MIS). For instance $\mathcal{I}$, an *independent set* $S \subseteq \mathcal{V}$ contains no adjacent node pair. MIS aims to find S that maximizes $|S|$ subject to $\forall i, j \in S, (i, j) \notin \mathcal{E}$.

6) Maximum Clique (MCl). A *clique* $K \subseteq \mathcal{V}$ is a node subset in which all pairs are connected by an edge. MCl aims to find K that maximizes $|K|$ subject to $\forall i, j \in K, i \neq j \Rightarrow (i, j) \in \mathcal{E}$.

7) Maximum Cut (MCut). A *cut* $C = (S, \bar{S})$ is a bipartition of $\mathcal{V}$ into disjoint sets S and $\bar{S}$. MCut aims to find C that maximizes $\sum_{i \in S, j \in \bar{S}} \mathbf{C}_{ij}$ with $\mathbf{C}$ being the adjacency matrix of $\mathcal{I}$.

B.3 Non-graph-based Tasks

8) Knapsack Problem (KP). Given a set of items $\mathcal{I} = \{1, 2, \cdots, M\}$, where each item $i \in \mathcal{I}$ is associated with two attributes: a weight $w_i \in \mathbb{R}_+$ and a value $v_i \in \mathbb{R}_+$, and a knapsack with a maximum weight capacity $W \in \mathbb{R}_+$, the objective is to select a subset $\mathcal{S} \subseteq \mathcal{I}$ of items such that the total weight of the selected items does not exceed the knapsack capacity, i.e., $\sum_{i \in \mathcal{S}} w_i \leq W$, while maximizing the total value of the selected items: $\max_{\mathcal{S} \subseteq \mathcal{I}} \sum_{i \in \mathcal{S}} v_i$. In this work, we focus exclusively on the **0-1 Knapsack Problem** that aligns with the settings in [11, 31, 32], where each item can be optionally included in the subset $\mathcal{S}$ only once.

9) Flexible Flow Shop Problem (FFSP). Given an instance $\mathcal{I}$ with N jobs and $S > 1$ ordered stages, each containing $M > 1$ parallel machines, let $\mathbf{D}_{ij}^{(k)}$ denote the processing time of job j on machine i at stage k. Each job must be processed without interruption on exactly one machine per stage and complete each stage before entering the next. Each machine can process at most one job at a time. Following [13], FFSP aims to minimize the makespan $C_{\max} = \max_{j=1,\ldots,N} C_j$, where C_j is the completion time of job j at the final stage.

C Additional Clarification on Our Positioning

C.1 Evaluation Scope Comparison within Related Works

This section is kindly intended for readers who are less familiar with the neural combinatorial optimization (NCO) literature. To help contextualize our evaluation protocol, Table 15 summarizes representative recent methods and the problem families and graph scales on which they are primarily evaluated.

Table 15: A comprehensive and comparative summary of recent and closely related NCO methods, including their learning paradigms, applicable tasks, and main evaluated problem *scopes* and *scales*. "Max Scale" denotes the largest evaluated graph size for main results (supplementary-only results are excluded). "–" denotes not applicable. Against this background, *our evaluation spectrum is well aligned with established practice in the broader NCO community, while being more extensive than most RL4CO baselines or prior methods that focus exclusively on either node- or edge-decision tasks.*

Method	Type	Evaluated *Edge* Tasks	Max Scale	Evaluated *Node* Tasks	Max Scale
Global, Non-autoregressive, Heatmap-guided methods (GP/AE)					
GCN4CO [15]	SL	TSP	100	–	–
GNNGLS [69]	SL	TSP	100	–	–
UnifyML4TSP [55]	SL	TSP	500	–	–
UTSP [68]	UL	TSP	100	–	–
Meta-EGN [35]	UL (Meta)	–	–	MIS, MCl, MVC	500
VAG-CO [34]	UL	–	–	MIS, MCl, MCut, MVC	300
DiffUCO [36]	UL (Gen)	–	–	MIS, MCl, MCut, MVC, MDS	1,200
RLSA [78]	sampling	–	–	MIS, MCl, MCut	11,000
FLD [79]	sampling	–	–	MIS, MCl, MCut	11,000
iSCO [77]	sampling	TSP	10,000	MIS, MCl	11,000
DIMES [33]	RL (Meta)	TSP	10,000	MIS	11,000
M^2GenCO [80]	SL (Meta, Gen)	TSP, ATSP	500	MIS, MCl, MCut	1,200
T2TCO [17]	SL (Gen)	TSP	1,000	MIS	800
Fast-T2T [37]	SL (Gen)	TSP	1,000	MIS	800
DIFUSCO [76]	SL (Gen)	TSP	10,000	MIS	800
COExpander [21]	SL (Gen)	TSP, ATSP, CVRP	10,000	MIS, MCl, MCut	3,000
ML4CO-Bench-101 [57]	SL (Gen)	TSP, ATSP, CVRP	10,000	MIS, MCl, MCut	3,000
Constructive, Autoregressive, Sequential-decision methods (LC)					
AM [10]	RL	TSPs, VRPs, OP	100	–	–
POMO [11]	RL	TSP, CVRP	100	–	–
MatNet [13]	RL	ATSP	100	–	–
Sym-NCO [12]	RL	TSPs, CVRP, OP	100	–	–
LCP [30]	RL	TSP, PCTSP, CVRP	100	–	–
MAB-MTL [84]	RL	TSP, CVRP, OP	100	–	–
MVMoE [85]	RL	VRPs	100	–	–
MTNCO [86]	RL	VRPs	100	–	–
BQ-NCO [31]	SL	TSP, ATSP, CVRP, OP	100	–	–
UNCO [87]	RL	TSP, CVRP	100	–	–
UniCO [24]	RL	TSP, ATSP	100	–	–
GOAL [32]	SL	ATSP, VRPs	100	MIS	100
UniteFormer [29]	RL	TSP, ATSP, CVRP	100	–	–
BOPO [88]	RL	TSP	100	–	–
PO4CO [70]	RL	TSP, CVRP	100	–	–
UCPO [89]	RL	TSPs, CVRPs	100	–	–
EFormer [28]	RL	TSP, CVRP	100	–	–
GREAT [27]	RL/SL	TSP, ATSP	100	–	–
ICAM [54]	RL	TSP, CVRP	1,000	–	–
RADAR [58]	RL	ATSP, ACVRP	1,000	–	–
GLOP [49]	D&C (RL)	TSP, CVRP, PCTSP	100,000	–	–
UDC [51]	D&C (RL)	TSP, ATSP, PCTSP, VRPs, OP	100,000	MIS	800
DualOPT [20]	D&C (RL)	TSP	100,000	–	–
Applicability across GP/LC/AE paradigms, edge/node problems, RL/SL/Generative learning objectives, and adaptable to D&C pipelines.					
UECO (ours)	RL/SL/Gen/D&C	TSP, ATSP, CVRP, ACVRP	100,000	MIS, MCl, MCut	11,000

Remark 1: Comparing within the Broad NCO Realm. As the table shows, our choice of tasks and instance sizes is well aligned with established practice in mainstream NCO research rather than being unusually narrow or conservative. In particular, *edge-based routing problems in recent RL-style NCO work are still predominantly evaluated at relatively modest scales (often around 100 nodes). Larger-scale results are more commonly reported in heatmap-guided or generative settings, but they mainly focus solely on routing problems and could hardly perform on node-based tasks*. On the node side, recent methods typically evaluate up to the hundreds or low thousands, depending on the problem family and paradigm. Against this backdrop, our evaluation spectrum is both representative and comparatively strong: it covers multiple major edge tasks (TSP, ATSP, and CVRP) up to scale 1000, and multiple major node tasks (MIS, MCl, and MCut) up to scale 3000. Therefore, our experimental setting is not only consistent with prevailing community standards, but also relatively broad and advanced in scope and scale, especially from the perspective of recent RL4CO-style evaluation.

Remark 2: Comparing with "Unified" Models. Recent works (e.g. UniteFormer [29]) also attempt to "unify" a GCN-style sub-encoder for edge features alongside a conventional MHA sub-encoder for node features. However, 1) node and edge information are processed and updated largely in separate, not integrated (until decoding) as a topological inductive bias in its entirety during inter-layer propagation, limiting their synergistic effect on representation learning; 2) the applicability is evaluated only on small-scale VRPs ($|\mathcal{V}| \leq 100$), and is solely adapted for RL-driven LC paradigms, leaving substantial room for both methodological advances in node-edge interactive modeling and empirical improvements for broader scalable validation.

C.2 Further Comparison against Edge-aware Attention Works for NCO

In addition to Table 1 which summarizes the comparison with recent edge-aware NCO architectures, we further position UECO against recent NCO architectures that explicitly exploit edge, matrix, or pairwise relational information. These works are highly relevant, but they differ in where edge information enters the architecture, whether edge states are dynamically refined, and whether the resulting encoder is intended to serve as a plug-in backbone across decision paradigms, learning objectives, and problem categories.

MatNet [13] is an important early step toward matrix-based NCO. Its mixed-score attention directly combines content-based attention scores with externally given matrix entries, enabling neural solvers for ATSP and FFSP where pairwise relations are primary inputs. **GREAT** [27] takes a different and complementary route: it elevates edges to first-class representation units and performs attention-based message passing among incident edges, which is particularly natural for edge classification, graph sparsification, and non-Euclidean routing. **EFormer** [28] further improves edge-only VRP solving by deriving temporary node embeddings from edge weights and then processing graph- and node-level representations in parallel. **UniteFormer** [29] extends this line by supporting node-only, edge-only, and hybrid input modalities within a single VRP solver, highlighting the practical importance of modality flexibility. **UniCO** [24] is orthogonal to these architecture-centric designs: it pursues problem-level unification by reducing different CO problems to matrix-encoded general TSP, and instantiates this framework with matrix-based RL and generative solvers. **COExpander** [21] is another relevant baseline in terms of task coverage rather than attention design. It proposes the Adaptive Expansion (AE) paradigm and supports both node-decision tasks and edge-decision tasks through a heatmap-based determination process. Nevertheless, its backbone is mainly a task-agnostic GCN-based global predictor, without mixed attention or edge-aware attention-score fusion.

Highlight. UECO is motivated by a different but related goal. Rather than specializing in a particular edge-only routing formulation, or unifying problems through reduction to a single matrix-form endpoint, UECO targets a plug-in encoder that can be substituted into different NCO pipelines. Its key distinction is the ***depth-wise closed loop between local structure-aware edge evolution and global node-centric attention***: the local module refines edge representations at each depth, and the resulting evolved edge states modulate the global attention scores, while the updated node semantics feed the next round of edge evolution. This design preserves the global relational modeling ability of Transformer-style attention, while avoiding a purely static use of raw edge features. As a result, UECO is applicable across LC, GP, and AE paradigms, across RL, SL, and generative objectives, and across both routing-style edge-decision tasks and sparse graph node-decision tasks. We view the above works as complementary advances in edge-aware NCO, while UECO contributes a backbone-level mechanism for dynamic local-global fusion and broad cross-paradigm reuse.

D Detailed Settings for Training and Solving

D.1 Hardware Configuration

The models for edge-decision tasks (TSP/ATSP/CVRP) with 50/100 nodes, and for node-decision tasks (MIS/MCl/MCut) with 200-300 nodes (and SATLIB) are trained on 4 NVIDIA 4090 (24GB) GPUs, while the models for edge-decision tasks with 200/500 nodes, and for node-decision tasks with 800-1200 nodes are trained on an NVIDIA A100 (80GB) GPU. All evaluations are conducted on a single NVIDIA 4090 (48GB) GPU with an Intel(R) Xeon(R) Platinum 8481C CPU, and are executed with a batch-size of 1 (consistent with previous works) without any multi-thread tricks.

D.2 Hyper-parameters and Model Settings

Main Experiments. We list core hyper-parameters for respective decision paradigms and problem types in the following tables. Throughout this work, an AdamW [90] optimizer is adopted with weight decay of 1e-5 for LC models and 5e-5 for GP/AE models.

Table 16: Hyper-parameters and datasets for training UECO with GP and AE paradigms under supervised learning on different COPs. Pre-train: checkpoint used for initialization. Epoch: minimum number of training epochs to reproduce reported results. LR: learning rate. Sparse: sparsification factor by k-NN (-1 indicates dense graph).

ID	Problem	Model	Pre-train	Training Parameters				Model Parameters		
				Data Size	Epoch	Batch Size	LR	Sparse	L_{global}	L_{local}
1	MCl	RB-[200-300]	–	64,000	50	8	2e-4	N/A	4	3
2	MCl	RB-[800-1200]	RB-[200-300]	6,400	10	2	5e-5	N/A	3	3
3	MCut	BA-[200-300]	–	128,000	50	8	2e-4	N/A	4	3
4	MCut	BA-[800-1200]	BA-[200-300]	128,000	10	4	5e-5	N/A	4	3
5	MIS	RB-[200-300]	–	64,000	50	8	2e-4	N/A	4	3
6	MIS	RB-[800-1200]	RB-[200-300]	6,400	10	2	5e-5	N/A	3	3
7	MIS	ER-[700-800]	–	12,800	50	4	2e-4	N/A	3	3
8	MIS	SATLIB	–	39,500	50	4	2e-4	N/A	4	3
9	TSP	Uniform-50	–	1,280,000	50	32	2e-4	-1	5	3
10	TSP	Uniform-100	Uniform-50	1,280,000	50	16	2e-4	-1	5	3
11	TSP	Uniform-500	Uniform-100	64,000	50	4	2e-4	50	5	3

Table 17: Hyper-parameters and datasets for training UECO with LC paradigm under reinforcement learning. Samples/Epoch: number of training instances per epoch. LR: learning rate. Sparse: sparsification factor by k-NN (-1 indicates dense graph).

ID	Problem	Model	Pre-train	Training Parameters				Model Parameters		
				Samples/Epoch	Epoch	Batch Size	LR	Sparse	L_{global}	L_{local}
1	TSP	Uniform-50	–	10000	8000	100	2e-4	-1	4	1
2	TSP	Uniform-100	Uniform-50	10000	3000	25	2e-4	-1	4	1
3	TSP	Uniform-500	Uniform-100	1000	1000	1	1e-4	-1	4	1
4	CVRP	Uniform-50	–	10000	3000	200	2e-4	-1	4	1
5	CVRP	Uniform-100	Uniform-50	10000	2000	50	2e-4	-1	4	1
6	CVRP	Uniform-200	Uniform-100	1000	1000	5	1e-4	-1	4	1
7	CVRP	Uniform-500	Uniform-200	10000	500	1	5e-5	-1	4	1
8	ATSP	Uniform-50	–	10000	8000	100	2e-4	-1	4	1
9	ATSP	Uniform-100	Uniform-50	10000	3000	25	2e-4	-1	4	1
10	ATSP	Uniform-200	Uniform-100	10000	1000	5	2e-4	-1	4	1
11	ATSP	Uniform-500	Uniform-200	1000	500	1	5e-5	-1	4	1

The hidden dimension d is always set as 256, with $h = 16$ attention heads. All other settings for baseline methods are kept as default following their official implementation or description in the original papers. Note that all models are trained with a curriculum learning manner to best accumulate advantages along the training process.

Additional ACVRP-100 Experiments. For the ACVRP-200 results in Table 12, we follow RADAR [58] and train for 2100 epochs, with 10,000 instances per epoch, under the LC paradigm with POMO policy sampling and a learning rate of 4×10^{-4}. As in the other experiments, we replace the encoder in RADAR with our UECO backbone trained under the LC paradigm, while keeping all remaining components unchanged, including SVD decomposition and distance-matrix construction. This isolates the effect of the backbone and highlights the expressiveness of UECO.

Additional TSP-1K to TSP-100K Experiments. For the TSP-1K to TSP-100K results in Table 8, we adapt our UECO model, trained under the LC paradigm, as a drop-in replacement for the original local sub-solvers in DualOPT [20], a state-of-the-art divide-and-conquer framework for large-scale TSP. For a fair comparison, we rerun DualOPT using its officially released checkpoints for the attention-based neural solvers. We keep the original DualOPT settings unchanged, using $N_{\text{iter}} = \{2, 3, 4\}$ grid-based searches for TSP-1K, TSP-10K, and TSP-100K, respectively, and path-refinement iterations $iter = \{25, 10, 5\}$.

E Supplementary Experimental Results

E.1 Full Results on TSP with 2OPT Enabled for Several Baselines

In Table 18, we provide full results on TSP, with additional results on several baselines (e.g. UTSP, DIMES, GNNGLS, DIFUSCO) where the 2OPT post-processing technique is enabled, as supplementary reference for Table 2 in the main text.

Table 18: Full results on TSP.

Method	Category	TSP-50 (1280 inst.)			TSP-100 (1280 inst.)			TSP-500 (128 inst.)			TSP-1000 (128 inst.)		
		Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
Learning-free Exact or Heuristics Solvers													
Concorde	Exact	5.69	0.000%	0.06s	7.76	0.000%	0.24s	16.55	0.000%	18.67s	23.12	0.000%	84.41s
LKH-3 (trails=500)	Heuristic	5.69	0.001%	0.06s	7.76	0.001%	0.18s	16.55	0.002%	1.85s	23.13	0.005%	4.64s
Global Prediction Methods													
UTSP (G)	GP-UL(OS)-GCN	7.10	24.808%	0.01s	9.38	20.909%	0.01s	19.76	19.435%	0.06s	N/A	N/A	N/A
UTSP (G + 2OPT)	GP-UL(OS)-GCN	5.98	5.080%	0.01s	8.16	5.215%	0.02s	17.71	7.033%	0.09s	N/A	N/A	N/A
DIMES (G)	GP-RL(OS)-GCN	6.33	11.233%	0.00s	8.76	12.918%	0.01s	18.89	14.155%	0.27s	26.82	16.013%	0.61s
DIMES (G + 2OPT)	GP-RL(OS)-GCN	5.89	3.578%	0.01s	8.11	4.543%	0.01s	17.66	6.707%	0.31s	24.73	6.953%	0.67s
GNNGLS (G)	GP-SL(OS)-GCN	6.18	8.651%	0.02s	8.97	15.597%	0.02s	–	–	–	–	–	–
GNNGLS (G + 2OPT)	GP-SL(OS)-GCN	5.71	0.333%	0.02s	7.86	1.295%	0.13s	–	–	–	–	–	–
GCN4CO (G)	GP-SL(OS)-GCN	5.72	0.520%	0.01s	7.94	2.334%	0.01s	18.88	14.098%	0.05s	28.02	21.188%	0.20s
GCN4CO (G + 2OPT)	GP-SL(OS)-GCN	5.69	0.066%	0.01s	7.78	0.243%	0.01s	16.85	1.863%	0.06s	23.97	3.703%	0.23s
UECO (G)	GP-SL(OS)-Uni.	5.70	0.147%	0.01s	7.81	0.635%	0.01s	18.24	10.259%	0.06s	26.61	15.125%	0.21s
UECO (G + 2OPT)	GP-SL(OS)-Uni.	**5.69**	**0.034%**	0.01s	**7.76**	**0.091%**	0.01s	**16.80**	**1.559%**	0.06s	**23.63**	**2.207%**	0.23s
DIFUSCO (G, I=50)	GP-SL(Gen)-GCN	5.71	0.384%	0.39s	7.85	1.136%	0.41s	18.14	9.639%	1.19s	–	–	–
DIFUSCO (G + 2OPT, I=50)	GP-SL(Gen)-GCN	5.69	0.105%	0.39s	7.78	0.261%	0.41s	16.80	1.524%	1.20s	–	–	–
Fast-T2T (G, I=1)	GP-SL(Gen)-GCN	5.72	0.612%	0.01s	7.94	2.381%	0.02s	18.72	13.128%	0.07s	26.78	15.854%	0.30s
Fast-T2T (G, I=5)	GP-SL(Gen)-GCN	5.69	0.061%	0.03s	7.77	0.230%	0.06s	18.58	12.280%	0.14s	26.57	14.918%	0.57s
Fast-T2T (G + 2OPT, I=5)	GP-SL(Gen)-GCN	5.69	0.029%	0.03s	7.76	0.078%	0.06s	16.86	1.903%	0.16s	23.67	2.393%	0.61s
UECO (G, I=1)	GP-SL(Gen)-Uni.	5.71	0.397%	0.01s	7.89	1.731%	0.02s	18.53	11.965%	0.07s	26.74	15.656%	0.23s
UECO (G, I=5)	GP-SL(Gen)-Uni.	5.69	0.026%	0.06s	7.77	0.140%	0.06s	18.30	10.615%	0.20s	26.55	14.766%	0.68s
UECO (G + 2OPT, I=5)	GP-SL(Gen)-Uni.	**5.69**	**0.014%**	0.06s	**7.76**	**0.053%**	0.06s	**16.79**	**1.489%**	0.21s	**23.55**	**1.845%**	0.69s
Local Construction Methods													
GOAL (G)	LC-SL(OS)-Trans.	5.74	0.965%	0.39s	7.91	1.920%	0.76s	18.01	8.825%	3.96s	25.38	9.789%	10.34s
AM (G)	LC-RL(OS)-Trans.	5.75	1.040%	0.01s	7.95	2.520%	0.03s	–	–	–	–	–	–
Sym-NCO (G)	LC-RL(OS)-Trans.	5.70	0.258%	0.05s	7.87	1.491%	0.10s	–	–	–	–	–	–
PreferenceCO (G)	LC-RL(OS)-Trans.	5.70	0.140%	0.03s	7.77	0.196%	0.01s	–	–	–	–	–	–
UniteFormer-XE (G)	LC-RL(OS)-Trans.	5.76	1.313%	0.22s	8.11	4.600%	0.43s	–	–	–	–	–	–
UniteFormer-XE (S=8)	LC-RL(OS)-Trans.	5.69	0.110%	0.23s	7.93	2.279%	0.47s	–	–	–	–	–	–
POMO (G)	LC-RL(OS)-Trans.	5.70	0.196%	0.04s	7.83	0.986%	0.05s	18.19	9.913%	0.23s	27.39	18.483%	1.14s
POMO (G + 2OPT)	LC-RL(OS)-Trans.	5.70	0.159%	0.04s	7.82	0.798%	0.05s	17.58	6.255%	0.55s	25.30	9.458%	6.13s
UECO (G)	LC-RL(OS)-Uni.	5.69	0.039%	0.06s	7.77	0.216%	0.06s	16.82	1.632%	0.35s	24.21	4.736%	1.27s
UECO (G + 2OPT)	LC-RL(OS)-Uni.	**5.69**	**0.035%**	0.06s	**7.77**	**0.193%**	0.06s	**16.78**	**1.445%**	0.42s	**23.86**	**3.192%**	2.95s
Adaptive Expansion Methods													
COExpander (G, D=3)	AE-SL(Gen)-GCN	5.70	0.244%	0.02s	7.85	1.164%	0.03s	18.77	13.447%	0.13s	26.81	15.966%	0.45s
COExpander (G + 2OPT, D=3)	AE-SL(Gen)-GCN	5.69	0.029%	0.02s	7.77	0.124%	0.03s	16.86	1.897%	0.15s	23.63	2.198%	0.50s
UECO (G, D=3)	AE-SL(Gen)-Uni.	5.69	0.089%	0.04s	7.79	0.411%	0.04s	18.53	11.989%	0.14s	26.76	15.763%	0.45s
UECO (G + 2OPT, D=3)	AE-SL(Gen)-Uni.	**5.69**	**0.014%**	0.04s	**7.76**	**0.066%**	0.04s	**16.77**	**1.330%**	0.15s	**23.53**	**1.773%**	0.49s

E.2 Additional Baseline Comparisons (INViT and L2C)

Table 19: Additional baseline comparison on TSP-100 with greedy decoding.

Method	Category	Obj.↓	Gap↓	Time
Concorde [64]	Exact	7.76	0.000%	0.24s
L2C [71]	LC-SL	7.79	0.460%	18.00s
INViT [72]	LC-RL	7.87	1.446%	3.47s
UECO	LC-RL	**7.77**	**0.216%**	**0.06s**

Remark. Despite the already broad set of baselines in our experiments, we remain open to further enriching the comparison and discussion. To this end, we made the effort to reproduce L2C [71] and INViT [72] under our taxonomy framework and additionally evaluated them on TSP-100 with greedy decoding; the results in Table 19 show that UECO still achieves the best overall trade-off between solution quality and efficiency.

Note that GenSCO [23] is also a highly promising concurrent work. However, we do not include it in our comparison because its positioning is different from our LC/GP/AE taxonomy–it is closer to a neuralized local-search or test-time scaling framework than to a standard (near) end-to-end solver in our setting, and its implementation is not natively aligned with our unified PyTorch framework, involving a substantially different engineering stack. We therefore view GenSCO as a complementary direction rather than a directly plug-in comparable baseline, and combining its search-driven refinement framework with our encoder-level contribution would be an interesting future direction.

E.3 Additional Results on ACVRP

In Table 20, we provide additional results on ACVRP. Following RADAR [58], we use the official open-sourced test datasets and set the test batch sizes to fully utilize one GPU card, i.e., 400 for ACVRP-100 and 100 for ACVRP-200. We plug UECO into the RADAR pipeline by replacing its encoder under the LC-RL setting while keeping the remaining protocol unchanged. These results show that UECO also transfers effectively to asymmetric capacitated routing and provides consistent gains as a plug-in encoder for more recent SOTA works under the same evaluation setup.

Table 20: More results on ACVRP. The baseline performance are quoted form RADAR [58], and our model is tested on the same datasets they officially open-source.

Method	**ACVRP-100** (1,000 inst.)			**ACVRP-200** (1,000 inst.)		
	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
LKH (100 trials)	2.2526	6.05%	2.20m	2.2245	2.77%	3.24m
LKH (1000 trials)	2.1635	1.86%	17.64m	2.1807	0.75%	25.32m
LKH (10000 trials)	2.1240	0.00%	2.79h	2.1645	0.00%	4.25h
HGS-Short	2.1614	1.76%	16.67m	2.0806	-3.88%	25.38m
HGS-Long	2.0942	-1.40%	2.78h	1.9733	-8.83%	4.24h
MatNet (Demand)	2.1968	3.42%	0.04m	3.1620	46.10%	0.20m
MatNet-Single (Demand)	2.1821	2.73%	0.03m	2.6283	21.43%	0.15m
MatNet-Single (Random)	2.1813	2.70%	0.03m	2.2372	3.36%	0.15m
ReLD	2.1656	1.96%	0.04m	2.1635	-0.05%	0.18m
RADAR	2.1596	1.67%	0.05m	2.1483	-0.75%	0.14m
UECO	**2.1587**	**1.62%**	0.08m	**2.1009**	**-2.94%**	0.35m

E.4 Additional Results on Cross-distribution Generalization (6 settings)

Table 21: Results for cross-distribution generalization. For ATSP-HCP-50, the oracle corresponds to a zero-cost Hamiltonian cycle, hence the relative optimality gap is mathematically undefined.

Method	**MIS-HK-[200–300]** (500 inst.)			**MCl-HK-[200–300]** (500 inst.)			**MCut-HK-[200–300]** (500 inst.)		
	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time
Oracle	111.474	0.000%	54.171s	4.390	0.000%	1.843s	721.934	0.000%	60.092s
UECO	109.816	1.488%	0.311s	3.898	10.620%	0.320s	689.210	4.551%	0.018s
Method	**TSP-Gaussian-50** (1280 inst.)			**TSP-Cluster-50** (1280 inst.)			**ATSP-HCP-50** (2500 inst.)		
	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
Oracle	23.840	0.000%	0.173s	3.729	0.000%	0.144s	0.000	–	0.110s
UECO	23.853	0.052%	0.131s	3.738	0.218%	0.132s	2.713	N/A	0.036s

Table 21 presents cross-generalization results on unseen distributions. Note that the models for TSP and ATSP are pretrained on uniform 2D data with 100 nodes, models for MIS and MCl are pretrained on RB graphs with 200-300 nodes, and the model for MCut is pretrained on BA graphs with 200-300 nodes, respectively. The test datasets are downloaded from the open benchmarks [57].

E.5 Additional Results on UECO with D&C Pipelines

Table 22: UECO as plug-in sub-solvers for the DualOPT [20] D&C frameworks for ultra-large TSP instances. Here, "len" denotes the sub-path length used in path-based refinement, and "iter" denotes the number of refinement iterations for each length. "Adaptive" denotes setting $iter$ as $\{10, 5, 3\}$ for TSP-1K, $\{6, 4, 2\}$ for TSP-10K, and $\{2, 2, 1\}$ for TSP-100K.

Methods	**Revise_len**	**Revise_iter**	**TSP-1K**		**TSP-10K**		**TSP-100K**	
			Obj.↓	Gap↓	Obj.↓	Gap↓	Obj.↓	Gap↓
DualOPT (AM)	{50, 20, 10}	{25, 10, 5}	23.743	2.703%	73.179	1.985%	230.129	1.710%
DualOPT-UECO	{50, 20, 10}	{25, 10, 5}	23.735	2.669%	73.167	1.969%	230.055	1.677%
DualOPT-UECO	{500, 100, 50}	Adaptive	23.271	0.660%	72.221	0.650%	232.078	2.571%
DualOPT-UECO	{500, 100, 50}	{25, 10, 5}	**23.271**	**0.661%**	**72.212**	**0.637%**	**229.763**	**1.548%**

We further evaluate UECO-LC as plug-in sub-solvers within the divide-and-conquer pipeline under different refinement settings. For a fair comparison at the original revision lengths $\{50, 20, 10\}$,

both DualOPT and DualOPT-UECO use the same 50-node pretrained LC checkpoint for all local revisers. Even under this constrained setting, replacing the vanilla attention-based sub-solvers with UECO consistently improves DualOPT across all scales. More importantly, our LC solver enables substantially larger revision lengths up to 500 nodes. With $\{500, 100, 50\}$, DualOPT-UECO achieves much stronger results on TSP-1K and TSP-10K, and further improves TSP-100K when using the original $\{25, 10, 5\}$ refinement schedule. These results highlight an additional benefit of UECO: it extends the practical scalability of RL-based neural sub-solvers, allowing the D&C framework to optimize larger sub-problems more effectively.

E.6 Per-instance Results on TSPLIB and CVRPLIB

Table 23: Detailed solution quality on TSPLIB instances with 50-200 nodes using models trained on uniform TSP-100 data. Baseline results are quoted from previous works [17, 37, 69] under their respective reported (best) settings.

Instances	AM	GCN	Learn2OPT	GNNGLS	DIFUSCO	T2T	Fast-T2T (w/ GS)	UniteFormer	UECO
eil51	16.767%	40.025%	1.725%	1.529%	2.82%	0.14%	0.00%	0.5728%	0.026%
berlin52	4.169%	33.225%	0.449%	0.142%	0.00%	0.00%	0.00%	1.3823%	0.004%
st70	1.737%	24.785%	0.040%	0.764%	0.00%	0.00%	0.00%	2.3088%	0.000%
eil76	1.992%	27.411%	0.096%	0.163%	0.34%	0.00%	0.00%	3.5368%	0.187%
pr76	0.816%	27.793%	1.228%	0.039%	1.12%	0.40%	0.00%	2.6541%	0.000%
rat99	2.645%	17.633%	0.123%	0.550%	0.09%	0.09%	0.00%	12.2852%	0.000%
kroA100	4.017%	28.828%	18.313%	0.728%	0.10%	0.00%	0.00%	6.0567%	0.064%
kroB100	5.142%	34.686%	1.119%	0.147%	2.29%	0.74%	0.65%	4.6043%	0.068%
kroC100	0.972%	35.506%	0.349%	1.571%	0.00%	0.00%	0.00%	4.6227%	0.000%
kroD100	2.717%	38.018%	0.866%	0.572%	0.07%	0.00%	0.00%	4.2331%	0.072%
kroE100	1.470%	26.589%	1.832%	1.216%	3.83%	0.27%	0.00%	5.5049%	0.178%
rd100	3.407%	50.432%	1.725%	0.003%	0.08%	0.00%	0.00%	1.2292%	0.000%
eil101	2.994%	26.701%	0.387%	1.529%	0.03%	0.00%	0.00%	2.7419%	0.000%
lin105	1.739%	34.902%	1.867%	0.606%	0.00%	0.00%	0.00%	12.6031%	0.320%
pr107	3.933%	80.564%	0.898%	0.439%	0.91%	0.61%	0.62%	5.9364%	0.000%
pr124	3.677%	70.146%	10.322%	0.755%	1.02%	0.60%	0.08%	2.1313%	0.000%
bier127	5.908%	45.561%	3.044%	1.948%	0.94%	0.54%	1.50%	9.6315%	0.000%
ch130	3.182%	39.090%	0.709%	3.519%	0.29%	0.06%	0.00%	3.1876%	0.002%
pr136	5.064%	58.673%	0.000%	3.387%	0.19%	0.10%	0.01%	3.5383%	0.000%
pr144	7.641%	55.837%	1.526%	3.581%	0.80%	0.50%	0.39%	1.6992%	0.176%
ch150	4.584%	49.743%	0.312%	2.113%	0.57%	0.49%	0.00%	3.7392%	0.000%
kroA150	3.784%	45.411%	0.724%	2.984%	0.34%	0.14%	0.00%	14.4228%	0.000%
kroB150	2.437%	56.745%	0.886%	3.258%	0.30%	0.00%	0.07%	9.3856%	0.062%
pr152	7.494%	33.925%	0.029%	3.119%	1.69%	0.83%	0.19%	9.3529%	0.968%
u159	7.551%	38.338%	0.054%	1.020%	0.82%	0.00%	0.00%	8.4383%	0.000%
rat195	6.893%	24.968%	0.743%	1.666%	1.48%	1.27%	0.79%	23.3120%	1.160%
d198	373.020%	62.351%	0.522%	4.772%	3.32%	1.97%	0.86%	42.5075%	1.663%
kroA200	7.106%	40.885%	1.441%	2.029%	2.28%	0.57%	0.49%	13.8043%	0.211%
kroB200	8.541%	43.643%	2.064%	2.589%	2.35%	0.92%	2.50%	18.1838%	0.416%
Mean	16.767%	40.025%	1.725%	1.529%	0.97%	0.35%	0.28%	8.0554%	**0.192%**

Following [17, 21, 37, 57, 69], we as well provide the solving metrics for every named instance in TSPLIB[2] and CVRPLIB[3] in Table 23 and Table 24 for more detailed comparison, further validating the out-of-distribution generalization of UECO on real-world benchmarks. On TSPLIB, although all models are trained only on uniform TSP-100 data, UECO achieves the best mean gap of 0.192%, outperforming the strong prior baselines Fast-T2T [37] 0.28% and DIFUSCO [76] 0.97%. On CVRPLIB-Set-X, UECO again attains the best overall performance, yielding the lowest mean objective 29.219 and mean gap 4.455%, improving over ML4CO-Bench-101 [57] (5.55%) as well as stronger margins against Sym-NCO [12] (7.98%) and UniteFormer [29] (21.62%). Taken together, these results indicate that the advantage of UECO is not limited to synthetic in-distribution settings, but transfers robustly to realistic benchmark instances with substantial distribution shift.

E.7 Memory and Runtime Overhead

Table 25 reports the practical memory and runtime overhead of UECO under matched TSP settings for two representative paradigms (LC-RL and GP-SL). Overall, the results show that UECO remains practically tractable across scales. In particular, although the inference cost increases with problem size, the empirical log-log scaling exponents remain close to linear (~ 1.12 for time and ~ 1.15 for memory), indicating that the complexity does not blow up quadratically in practice. This observation

[2] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95

[3] http://vrp.atd-lab.inf.puc-rio.br/media/com_vrp/instances

Table 24: Detailed solution quality on CVRPLIB-Set-X instances with 0–200 nodes using models trained on uniform CVRP-100 data. Baseline results are quoted from previous works [57].

Instances	Optimal Obj.	Sym-NCO		UniteFormer		ML4COBench-101		UECO	
		Obj.↓	Gap↓	Obj.↓	Gap↓	Obj.↓	Gap↓	Obj.↓	Gap↓
X-n101-k25	27.905	30.625	9.744%	30.384	8.884%	30.563	9.524%	29.543	5.870%
X-n106-k14	26.362	26.969	2.300%	28.484	8.047%	26.802	1.668%	26.752	1.480%
X-n110-k13	14.979	15.097	0.784%	17.531	17.035%	15.097	0.784%	15.280	2.009%
X-n115-k10	12.933	13.896	7.448%	16.348	26.407%	13.574	4.957%	12.975	0.329%
X-n120-k6	13.437	13.611	1.296%	18.657	38.852%	13.611	1.296%	13.770	2.481%
X-n125-k30	56.164	59.041	5.122%	59.365	5.698%	59.041	5.122%	60.803	8.259%
X-n129-k18	29.005	29.476	1.625%	30.738	5.977%	29.356	1.211%	29.239	0.809%
X-n134-k13	11.351	11.633	2.479%	12.649	11.432%	11.633	2.479%	11.444	0.817%
X-n139-k10	13.678	14.048	2.708%	19.622	43.457%	13.804	0.920%	13.742	0.464%
X-n143-k7	15.776	16.533	4.798%	19.141	21.329%	16.533	4.798%	16.143	2.327%
X-n148-k46	43.671	53.319	22.092%	46.578	6.656%	45.190	3.478%	46.025	5.391%
X-n153-k22	21.572	31.742	47.146%	26.197	21.437%	28.809	33.547%	25.295	17.257%
X-n157-k13	17.572	17.708	0.775%	25.587	45.616%	17.708	0.775%	17.701	0.735%
X-n162-k11	14.412	14.705	2.029%	20.890	44.942%	14.705	2.029%	14.653	1.670%
X-n167-k10	20.766	21.244	2.303%	24.844	19.639%	21.244	2.303%	21.053	1.381%
X-n172-k51	46.974	65.767	40.007%	50.290	7.059%	59.677	27.041%	50.193	6.851%
X-n176-k26	47.816	51.999	8.748%	55.719	16.528%	51.999	8.748%	59.590	24.624%
X-n181-k23	26.809	27.067	0.963%	33.408	24.615%	27.067	0.963%	27.111	1.125%
X-n186-k15	24.251	24.720	1.933%	29.459	21.475%	24.720	1.933%	24.456	0.843%
X-n190-k8	17.037	17.817	4.580%	26.928	58.054%	17.418	2.238%	17.521	2.842%
X-n195-k51	44.274	45.772	3.385%	48.830	10.291%	45.772	3.385%	47.407	7.078%
X-n200-k36	60.084	62.080	3.324%	67.460	12.276%	61.834	2.914%	62.111	3.375%
Mean	27.583	30.221	**7.98%**	32.232	**21.62%**	29.371	**5.55%**	**29.219**	**4.455%**

Table 25: Memory and runtime overhead of UECO under representative and identical TSP settings. Training batch sizes follow Appendix D, and the test batch size is fixed to 1.

Phase	Method	Instance scale		
		50	100	500
Training	UECO (LC-RL)	20.14 GB 125 ms/step	21.08 GB 202 ms/step	45.6 GB 970 ms/step
	UECO (GP-SL)	20.8 GB 497 ms/step	13.2 GB 322 ms/step	17.1 GB 304 ms/step
Inference	UECO (LC-RL)	0.2 GB 27 ms/inst.	0.2 GB 46 ms/inst.	2.6 GB 359 ms/inst.
	UECO (GP-SL)	0.8 GB 32 ms/inst.	1.4 GB 37 ms/inst.	7.0 GB 96 ms/inst.

is consistent with the paper's broader efficiency-quality narrative: UECO is introduced as a unified encoder for fair cross-paradigm evaluation, and the main experiments already show that it achieves strong solution quality while maintaining competitive solving efficiency. Finally, the supplementary efficiency analysis further shows that runtime grows smoothly as model capacity increases, suggesting that UECO offers a favorable and tunable performance-efficiency trade-off for practitioners with different computational budgets.

E.8 In-depth Hierarchical Ablations: Whether It Works, What Matters, and Why?

Remark on Mechanistic Evidence. To directly test our central claim on ***iterative edge evolution*** and ***structure-aware mixed attention***, we conduct two focused LC-RL ablations (100 epochs): Table 26 freezes edge states at different depths, and Table 27 modifies the attention score to isolate the contribution of edge-aware structural scoring. This design closely follows the intended UECO mechanism, where the local module iteratively updates edge representations $\mathbf{G}^{(\ell)}$ and the mixed-

Table 26: Freezing edge states at different depths under LC-RL (100 epochs). Freezing at earlier depths degrades performance more clearly, whereas freezing at layer L remains close to the full model since most iterative evolution has already occurred. Note: Freeze-Layer-L should be interpreted as a weak perturbation that preserves almost all preceding edge evolution, so its closeness to the full model is expected and does not contradict the necessity of iterative refinement.

Task	Metric	Full	Freeze-Layer-0	Freeze-Layer-$L/2$	Freeze-Layer-L	$\Delta(0$ vs. Full$)$
TSP	Obj.↓	5.75	5.78	5.80	5.74	+0.03
	Gap(%)↓	1.03	1.56	1.93	1.00	+0.53
ATSP	Obj.↓	1.60	1.68	1.61	1.61	+0.08
	Gap(%)↓	2.75	7.82	3.61	3.40	+5.06
CVRP	Obj.↓	10.81	10.83	10.82	10.83	+0.02
	Gap(%)↓	4.31	4.48	4.35	4.44	+0.18

Table 27: Attention-score ablation under LC-RL (100 epochs). "Static-Edge" uses edge information without iterative depth-wise refinement; "No-Edge" removes structural scoring from the attention computation.

Task	Metric	Full	Static-Edge	No-Edge	Δ(No-Edge vs. Full)
TSP	Obj.↓	5.75	5.81	5.80	+0.05
	Gap(%)↓	1.03	2.06	1.94	+0.91
ATSP	Obj.↓	1.60	1.67	4.12	+2.52
	Gap(%)↓	2.75	7.47	165.09	+162.33
CVRP	Obj.↓	10.80	10.80	10.83	+0.03
	Gap(%)↓	4.16	4.14	4.44	+0.28

Table 28: Gap(%)↓ comparison on testing size-50 problems when removing the two core mechanisms as modules. "Raw POMO" [11] is included as a reference baseline backbone.

Variant	TSP (LC-RL)	ATSP (LC-RL)	CVRP (LC-RL)	TSP (GP-SL)
UECO (Full)	**0.04**	**0.32**	**0.45**	**0.01**
w/o Edge Evol.	0.74	9.13	1.62	16.83
w/o Mixed-Attn	4.16	24.46	1.89	8.14
Raw POMO	0.71	24.12	1.59	8.47

Table 29: Dual-channel ablation under LC-RL (100 epochs, 50 nodes). "w/o Dual-Channel" removes the dual-channel design. Gap differences are reported in percentage points.

Task	Metric	Full	w/o Dual-Channel	Δ(w/o vs. Full)
TSP	Obj.↓	5.726	5.730	+0.004
	Gap(%)↓	0.667	0.752	+0.085
ATSP	Obj.↓	1.595	4.201	+2.606
	Gap(%)↓	2.511	170.810	+168.299
CVRP	Obj.↓	10.798	10.805	+0.007
	Gap(%)↓	3.612	3.683	+0.071

attention block fuses content-based and structure-based scores into $\mathbf{S}_{\text{mix}}^{(\ell)}$, thereby forming a closed loop between node semantics and edge evolution. In other words, these supplementary tests do not merely remove parameters; they perturb the exact information path claimed in Secs. 3.2– 3.3.

Table 30: Plug-in gains of UECO on routing tasks under LC-RL. Metric: Gap (%). Baseline is POMO [11] for TSP/CVRP and MatNet [13] for ATSP. "Rel. red." is short for "relative reduction".

Task	TSP-50	TSP-100	TSP-500	ATSP-50	ATSP-100	ATSP-500	CVRP-50	CVRP-100	CVRP-500	Overall
Baseline	0.16	0.80	6.26	0.32	1.59	N/A	3.52	3.00	N/A	–
Ours (LC-RL)	0.04	0.19	1.73	0.32	0.65	2.84	0.45	0.86	1.84	–
Δgap	-0.12	-0.61	-4.53	-0.00	-0.94	N/A	-3.07	-2.14	N/A	-1.63
Rel. red. (%)	77.99	75.81	72.39	0.63	59.27	N/A	87.16	71.37	N/A	63.52

Table 31: Plug-in gains of UECO on node-decision tasks across paradigms. Metric: Gap (%).

Task	MIS-small	MIS-large	MCl-small	MCl-large	MCut-small	MCut-large	Overall
Baseline [15]	9.41	16.04	5.78	4.33	3.10	3.74	–
Ours (GP-SL)	7.68	15.83	3.35	7.19	2.54	3.49	–
Δgap	-1.74	-0.21	-2.44	+2.86	-0.56	-0.25	-0.39
Rel. reduction (%)	18.38	1.31	42.04	-66.05	18.06	6.68	3.40
Baseline [21]	2.09	4.06	1.89	8.82	0.06	-1.39	–
Ours (UECO-AE-SL)	1.26	2.78	1.04	3.48	0.01	-1.51	–
Δgap	-0.83	-1.29	-0.86	-5.34	-0.04	-0.12	-1.41
Rel. reduction (%)	39.56	31.65	45.24	60.53	75.86	8.72	43.59

At the micro level, Table 26 shows that freezing edge states at layer 0 or $L/2$ consistently hurts performance, with the largest degradation on ATSP (gap increasing from 2.75% to 7.82% when freezing from layer 0), which is fully consistent with UECO's design for directional and structurally sensitive problems. By contrast, freezing at layer L stays very close to the full model, which is also expected: by that point, the preceding iterative evolution has already been completed, so the perturbation is much weaker. Table 27 further shows that removing structural scoring from attention is especially destructive on ATSP (gap rising from 2.75% to 165.09%), while on TSP it also causes a clear drop. For CVRP, the effect is milder under the present 100-epoch budget, and the displayed gap of "Static-Edge" is even slightly better than "Full" (4.14 vs. 4.16), so the most accurate interpretation is not that mixed attention dominates uniformly everywhere, but that it is most critical when pairwise structure and asymmetry are central.

At the medium level, Table 28 shows that removing either core mechanism causes substantial degradation relative to the full UECO encoder. This is particularly evident on TSP (GP-SL), where the gap jumps from 0.01% to 16.83% without edge evolution and to 8.14% without mixed attention, indicating that the two modules are not cosmetic but functionally necessary when UECO serves as a plug-in encoder. On the LC-RL side, the full model also clearly outperforms Raw POMO on TSP and ATSP, while the ablated variants partially regress toward the backbone baseline.

At the high level, the plug-in summaries in Tables 30–31 show that these mechanism-level effects translate into broad cross-paradigm gains. The routing-side summary is uniformly positive, with an overall relative gap reduction of 63.52%. For node tasks, the AE-SL/COExpander pairing is consistently improved in all listed settings, whereas the GP-SL/GCN4CO pairing improves in five of the six displayed entries but degrades on MCl-large; hence the strongest accurate conclusion is that UECO yields *broad rather than literally universal* plug-in gains in the summarized subset. Taken together, the evidence is coherent across the micro, medium, and high levels, and supports the paper's mechanism claim that performance gains arise from *iterative edge refinement together with mixed structural attention*, rather than from an isolated empirical fluctuation.

E.9 Results for Statistical Stability

Stability is a critical yet often overlooked aspect in neural combinatorial optimization. We present box plots of objective values obtained by UECO and the corresponding optimal solutions over test instances. The two distributions exhibit highly similar medians and variances, indicating that UECO delivers stable and reliable performance across instances of varying difficulty. This consistency further confirms the robustness of the proposed method.

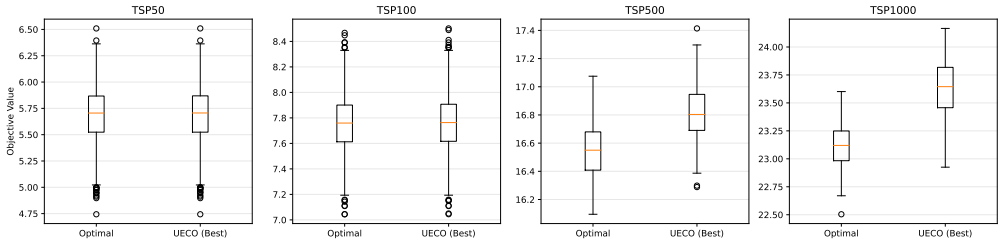


Figure 9: Box plots of the objective values obtained by UECO compared to the optimal label solutions on TSP benchmarks.

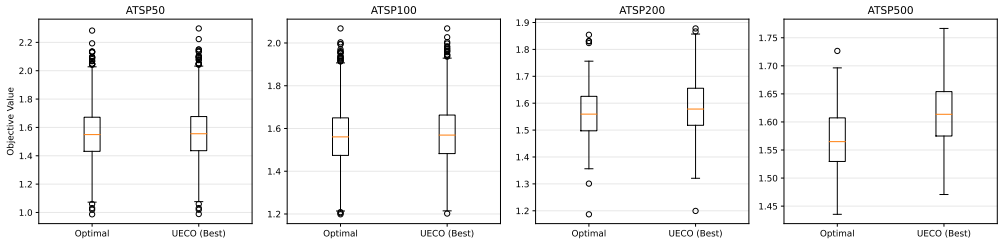


Figure 10: Box plots of the objective values obtained by UECO compared to the optimal label solutions on ATSP benchmarks.

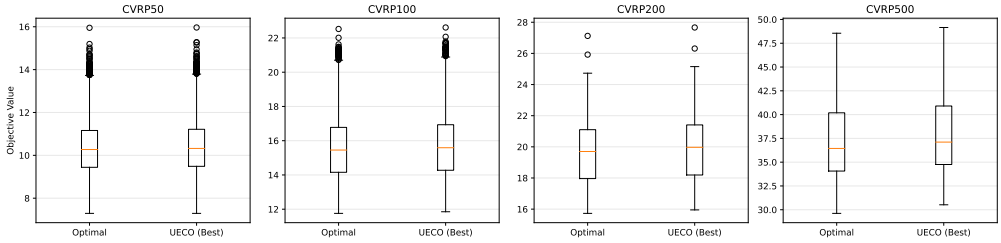


Figure 11: Box plots of the objective values obtained by UECO compared to the optimal label solutions on CVRP benchmarks.

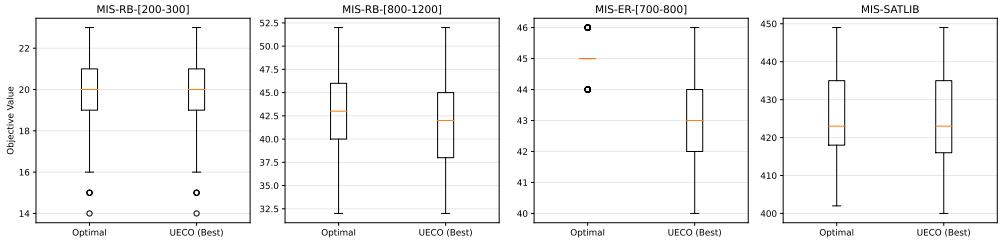


Figure 12: Box plots of the objective values obtained by UECO compared to the optimal label solutions on MIS benchmarks.

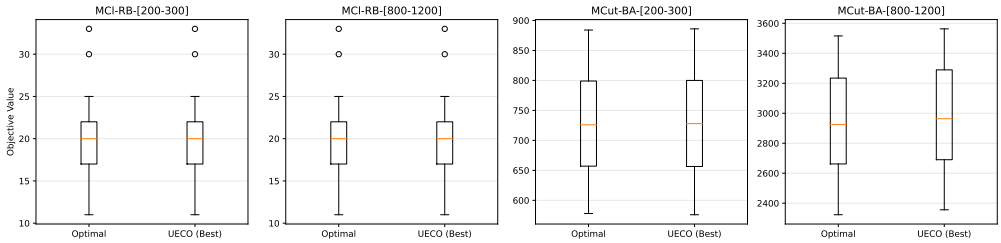


Figure 13: Box plots of the objective values obtained by UECO compared to the optimal label solutions on MCl and MCut benchmarks.

F Additional Observations and Discussions

F.1 Rethinking LC4CO: Towards Competitive Scalability upon UECO

As discussed in the main analysis, our results demonstrate that mainstream LC-based methods, when equipped with UECO as the backbone encoder, can achieve substantially improved learning efficiency and solution quality. This observation challenges the long-standing perception that RL-based routing solvers inherently suffer from poor scalability and unstable optimization [21, 55]. By decoupling representation learning from decision-making and strengthening the expressive capacity of the encoder, UECO significantly alleviates these limitations.

Additionally, as shown previously, LC solvers naturally integrate with state-of-the-art divide-and-conquer frameworks [16, 20, 49, 51]. For ultra-large instances, e.g., TSP with over 10,000 nodes, their autoregressive decision process makes them particularly suitable as local sub-solvers: they can flexibly handle decomposed sub-problems of varying sizes and granularities. In this sense, LC solvers offer a more adaptable route to large-scale solving than GP solvers, whose one-shot heatmap prediction is less naturally aligned with controllable sub-problem resolution. This further motivates our design of UECO as a stronger backbone for LC solvers.

Looking forward, recent advances such as preference-based optimization [70, 88, 89] have shown promise in addressing intrinsic challenges of RL, including sparse rewards and slow convergence in long-horizon decision processes. These approaches introduce alternative optimization objectives that better align training signals with solution quality. We view such methods as highly complementary to our work: while they focus on improving the *training objective*, UECO provides a strong and general *architectural foundation*. Combining preference-based learning schemes with our unified encoder presents a promising direction for future research, potentially enabling LC-based solvers to achieve both high scalability and strong optimality guarantees.

F.2 Diverged Efficacy of Post-inference Improving Techniques across Paradigms

As 2OPT is a lightweight and widely adopted post-inference refinement operator, we further concentrate on its paradigm-specific performance. Table 2 reveals a clear paradigm-dependent behavior of 2OPT. For GP solvers, 2OPT yields substantial and consistent gains. For instance, GCN4CO improves from $0.520\% \rightarrow 0.066\%$ on TSP-50 and $2.334\% \rightarrow 0.243\%$ on TSP-100, and the same trend holds for our backbone: UECO (G) reduces from $0.147\% \rightarrow 0.034\%$ on TSP-50, $0.635\% \rightarrow 0.091\%$ on TSP-100, and $10.259\% \rightarrow 1.559\%$ on TSP-500. Similar sensitivity appears for generative modeling, e.g., UECO (GP-SL(Gen)-Uni) and UECO (AE-SL(Gen)-Uni), etc.

In contrast, LC solvers exhibit noticeably weaker gains from the same post-processing. While 2OPT can still improve the final solution (e.g., POMO on TSP-100: $0.986\% \rightarrow 0.798\%$; UECO on TSP-100: $0.216\% \rightarrow 0.193\%$), the magnitude is typically smaller than in GP/AE, and the improvements are less consistent across scales (e.g., UECO on TSP-50: $0.039\% \rightarrow 0.035\%$, marginal; on TSP-500: $1.632\% \rightarrow 1.445\%$, moderate). We speculate that LC policies, trained to construct tours autoregressively, already internalize stronger neighborhood-level optimality during decoding, thereby leaving less exploitable local slack for a post-hoc 2OPT operator. By comparison, GP/AE predictions often provide globally coherent but locally imperfect structures (e.g., suboptimal edge exchanges), making them more amenable to local rewriting.

These observations highlight a broader methodological implication: post-inference improvement is not universally "plug-and-play" across paradigms, but interacts with *where* local and global regularities are leveraged (during learning, decoding, or after decoding). A promising future direction is to explicitly balance this division of labor: for example, designing training objectives or decoding strategies that preserve global exploration while intentionally deferring certain local refinements to efficient post-processing, or conversely, enhancing LC-style decoders with mechanisms that promote global consistency beyond greedy autoregressive commitments.

Notably, the modular interleaving of global attention and local topology evolution in UECO naturally exposes well-structured intermediate representations, making it especially amenable to downstream improvement operators. This suggests that architectural choices at the encoder level can fundamentally shape how effectively post-inference optimization can be exploited, opening up new directions for jointly designing unified backbones and refinement mechanisms.

G Limitation and Future Work

While this work focuses on a broad and representative class of graph-based COPs, covering most mainstream NCO benchmarks across routing and graph-selection tasks, several directions remain open for further exploration. First, the unification achieved by UECO is mainly at the encoder level: different problem families may still require task-specific decoders, feasibility masks, or lightweight output heads to respect their combinatorial constraints. Second, although UECO can be effectively combined with sparsification and divide-and-conquer strategies for large-scale instances, its global attention component may introduce additional computational cost on dense graphs, motivating future studies on more memory-efficient attention or adaptive neighborhood selection. Third, the supplementary KP and FFSP results provide initial evidence beyond graph-based tasks; extending UECO to broader problem families, including additional job scheduling problems (JSPs), remains a promising direction. We believe these future developments will further demonstrate the generality and flexibility of the unified encoder, and may enable more efficient learning and inference across an even broader spectrum of combinatorial optimization tasks.

H Broader Impact

This paper presents a unified encoder for Neural Combinatorial Optimization (NCO), with the goal of improving the generality and comparability of learning-based solvers across combinatorial problems and decision paradigms. Potential positive impacts include more efficient and reusable optimization methods for domains such as logistics, network design, scheduling, and scientific discovery. We do not identify any severe negative societal impacts specific to this work. The paper studies a generic encoder architecture on standard benchmark tasks and does not introduce a system intended for direct deployment in safety-critical settings. As with other optimization and machine learning methods, downstream applications may have mixed consequences depending on context, but such risks are indirect and not unique to our contribution. Overall, we believe the societal risk of this work is limited.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction accurately reflect the paper's contributions and scope, which are further validated in the experiment section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes. We discuss the limitations in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe in detail our proposed model architecture and algorithms, and provide sufficient hyper-parameter settings, hardware configurations, datasets used, and other model settings for reproducibility. Full codebase and datasets have been made publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Full code base and datasets have been made publicly available (https://github.com/Thinklab-SJTU/UECO). Besides, we describe in detail our proposed model architecture and algorithms, and provide sufficient hyper-parameter settings, hardware configurations, datasets used, and other model settings for reproducibility.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We describe in detail our proposed model architecture and algorithms, and provide sufficient hyper-parameter settings, hardware configurations, datasets used, and other model settings for reproducibility. Please see Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We have included statistical results in Appendix E.9.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please see Appendix D and Appendix E.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conform strictly with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please see Appendix H.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use existing public benchmark assets only for academic evaluation. The in-distribution training and testing datasets are adopted from ML4CO-Bench-101 [57], which is released under CC-BY 4.0. For generalization and real-world evaluation, we additionally use established public benchmarks including TSPLIB, CVRPLIB, SATLIB, TWITTER, and COLLAB, all under their non-commercial research-use terms with the required copyright and permission notices preserved.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Source code is available at `https://github.com/Thinklab-SJTU/UECO`, with documentation of the proposed models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components. LLMs are only used for polishing the writing.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.