NLD4CO: Neural Langevin Dynamics for Combinatorial Optimization

Jiale Ma[12], **Wenzheng Pan**[1†], **Binghao Cai**[1], **Xihe Zhang**[1], **Junchi Yan**[12*]
[1]Sch. of Computer Science & Sch. of Artificial Intelligence, Shanghai Jiao Tong University
[2]Shanghai Innovation Institute.
{heatingma,pwz1121,cbh10086,sjtuxihe,yanjunchi}@sjtu.edu.cn

Abstract

Langevin Dynamics (LD) provides a principled framework for solving combinatorial optimization problems (COPs) via gradient-guided stochastic search. However, classical LD faces two bottlenecks: 1) the utilization of uniform initialization may affect its convergence speed; 2) when applied to COPs such as routing problems that cannot be naturally formulated as Quadratic Unconstrained Binary Optimization (QUBO), classical LD suffers from slow convergence and inferior performance compared to existing learning-to-construct (L2C) methods, as it requires manually designed discrete proposals for updates. These proposals are typically local and induce abrupt transitions in the solution space, hindering escape from local optima and degrading performance. To address these issues, we propose NLD4CO, a unified framework that synergizes data-driven learning with LD. It introduces two instantiations: explicit-gradient (EG) and implicit-gradient (IG). NLD-EG accelerates sampling efficiency for energy-based problems via neural warm-start initialization. NLD-IG utilizes the corrective direction predicted by a consistency model as an implicit score function to guide the search, making structured, globally coordinated transitions for general COPs. Extensive evaluations on MIS, MWIS, TSP, and CVRP demonstrate that NLD4CO achieves SOTA performance, delivering superior solution quality with competitive or lower runtime.

1 Introduction

Combinatorial optimization problems (COPs) are fundamental to operations research and computer science, yet their NP-hard nature makes finding optimal solutions intractable at scale [1, 2]. While traditional solvers like Gurobi [3] and Concorde [4] guarantee optimality, their exponential time complexity limits practical applicability. Recently, neural combinatorial optimization (NCO) has shown strong potential for solving COPs [5, 6], and existing methods can be broadly categorized into: learning-to-construct (L2C), learning-to-search (L2S), and divide-and-conquer (D&C).

Although L2C methods [5–24] have achieved strong performance within a short time, their advantage diminishes as the problem scale increases. In particular, when relying solely on standard greedy decoding, these methods struggle to maintain solution quality, gradually losing their competitiveness against learning-free constructive approaches, as demonstrated in Appendix C. As for D&C methods [25, 26], while effectively reducing the problem scale, they are often constrained by the degradation in solution quality introduced during the decomposition step. Given the limitations of the two pipelines, we argue that L2S offers a particularly promising direction, as it can naturally incorporate existing L2C methods as initialization strategies, thereby benefiting from high-quality starting solutions. Moreover, its iterative refinement nature provides a flexible mechanism to balance

*Correspondence to: Junchi Yan. †W. Pan is a visiting student at Shanghai Innovation Institute. This work was partly supported by NSFC (92370201). Code is available at https://github.com/Thinklab-SJTU/NLD4CO.

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

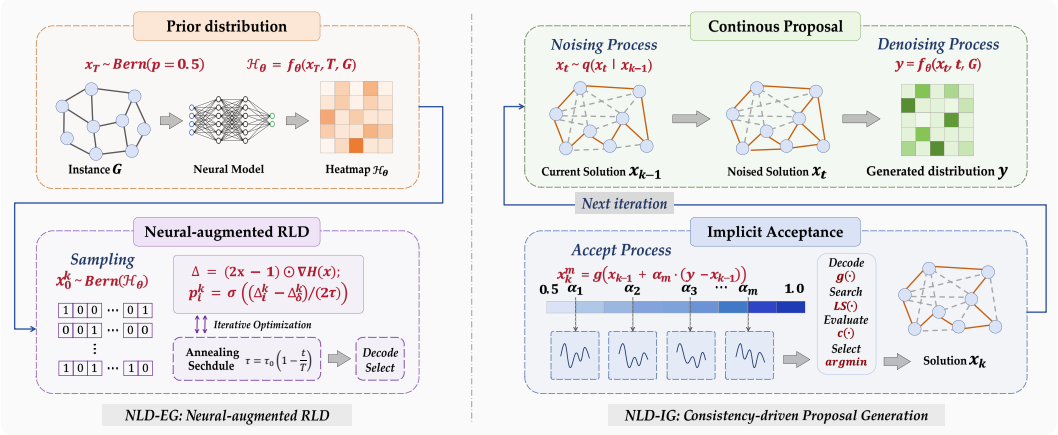

Figure 1: Overview of the NLD4CO framework. The two instantiations provide complementary neural guidance: NLD-EG learns an initialization distribution (left), while NLD-IG learns transition dynamics (right). 1) NLD-EG: $\mathcal{H}_\theta$ denotes the probability heatmap; $H(\cdot)$ the energy function; τ_0 the initial temperature; δ the regularization term. 2) NLD-IG: t is the diffusion timestep; $q(\cdot)$ the noising operator; α the interpolation coefficient; $g(\cdot)$ the decoding function; $c(\cdot)$ the cost.

the trade-off between solution quality and computational cost by adjusting the number of search steps or the time budget. Additionally, L2S can be further integrated with D&C schemes in certain scenarios [27, 28], enabling complementary strengths across different pipelines.

Existing L2S approaches primarily rely on neighborhood search [29–31], stochastic sampling [32], and perturb-and-optimize strategies [33]. Despite their diverse formulations, these methods share a common objective: escaping local optima to discover improved solutions. This objective naturally connects L2S to classical stochastic search paradigms, where Markov Chain Monte Carlo (MCMC) provides a principled foundation. Recently, Langevin Dynamics (LD) [34, 35], a gradient-guided variant of MCMC, has emerged as a promising approach for energy-based problems that can be formulated as Ising models [36, 37] or Quadratic Unconstrained Binary Optimization (QUBO). However, directly applying classical LD to COPs presents several challenges. For problems with explicit energy functions, LD typically applies a uniform random initialization, resulting in prohibitively long mixing times before entering high-quality regions. For problems without a natural QUBO formulation, such as routing tasks, designing effective discrete update operators is inherently challenging. Existing adaptations often rely on local, manually designed proposals (e.g., 2-OPT variants), which induce abrupt transitions and struggle to escape deep local minima.

To address the aforementioned limitations, we propose **Neural Langevin Dynamics for Combinatorial Optimization (NLD4CO)**, a generalized framework that integrates classical LD with data-driven learning, as sketched in Fig. 1. Recognizing that LD already performs competitively on energy-based formulations (e.g., Ising or QUBO models), we introduce only minimal modifications to preserve its strengths, ***while focusing our main contribution on extending its applicability to broader COPs.*** Specifically, for problems with explicit energy functions, the framework reduces to **NLD-EG (Explicit Gradient)**. In this setting, classical discrete LD is largely preserved, and we incorporate a lightweight neural initialization module to provide high-quality warm starts. This enhances mixing efficiency without altering the original gradient-driven dynamics. More importantly, for general COPs that lack explicit energy formulations (e.g., routing problems, such as TSP and CVRP), we propose **NLD-IG (Implicit Gradient)**. This variant departs from the standard discrete search paradigm by lifting solutions into a continuous latent space. Departing from classical LD that relies on explicit energy gradients, NLD-IG utilizes a learned proposal generator (instantiated by a consistency model [13, 38, 39] in this work) to produce corrective directions that guide the search. We then generate multiple candidate solutions by controlling the strength of these updates, decode them back to the discrete domain, and optionally refine them via local search. To provide a clearer and unified view, we compare MCMC, classical LD, and our proposed variants in Table 1.

The highlights include: (1) We propose NLD4CO, a unified framework that extends classical LD to a broad class of COPs. Unlike prior approaches that are limited to energy-based formulations, our framework generalizes LD to settings without explicit energies, significantly expanding its

Table 1: Comparison of Search-based Methods (MCMC, LD, NLD-EG, and NLD-IG).

METHOD	PROBLEMS	INITIALIZATION	PROPOSAL	ACCEPTANCE	SPACE
MCMC [43]	Energy-based COPs (MIS, MWIS)	Uniform Bernoulli	Stochastic	SA / MH	Discrete
LD [32, 42, 48]	Energy-based COPs (MIS, MWIS)	Uniform Bernoulli	Guided by Explicit Gradient	SA / MH	Discrete
NLD-EG (Ours)	Energy-based COPs (MIS, MWIS)	Neural Warm-start	Guided by Explicit Gradient	SA / MH	Discrete
NLD-IG (Ours)	General COPs (TSP, CVRP)	Construction-based	Guided by Implicit Gradient	Deterministic Selection	Discrete and Continuous Latent

applicability. (2) We develop a suite of learning-free baseline solvers (construction-based and search-based, Appendix B), providing standardized and reproducible reference points for fair comparison and future research. (3) We design fair evaluation protocols that align runtime budgets across different solver types (Appendix F). Under these controlled settings, NLD4CO achieves SOTA results on MIS, MWIS, TSP, and CVRP, while maintaining competitive or lower runtime.

2 Related Work

L2S Solvers. NeuOpt-GIRE [29] incorporates Tabu Search [40] to enable exploration in infeasible regions. However, it suffers from relatively slow speed and limited scalability to larger instances. RLD [32] leverages a regularization term to control the sampling distance from the current solution, ensuring more robust exploration. While effective for problems that can be formulated as Ising or QUBO models, it is less applicable to other domains such as routing problems. Recently, GenSCO [33] utilizes a rectified flow model to refine stochastically perturbed solutions. However, its performance depends heavily on the training distribution and degrades significantly under scale or distribution shifts, as noted in [41] and further confirmed by our analysis in Appendix C.

Generative Models. In recent years, generative models have increasingly shown strong promise for solving COPs. DIFUSCO [6] pioneers this direction by introducing supervised diffusion models that iteratively denoise corrupted solutions into high-quality ones. DiffUCO [14] further extends this framework by incorporating energy functions, enabling unsupervised training of diffusion models. To improve the efficiency of diffusion-based solvers, Fast-T2T [13] employs consistency training to significantly reduce the number of denoising steps during inference. GenSCO [33] takes a step further by unifying generative modeling with a search framework, where the current solution is stochastically perturbed and then regenerated via a flow-based model.

Markov Chain Monte Carlo. MCMC provides a principled stochastic framework for exploring the discrete solution spaces of COPs. By constructing Markov chains whose stationary distributions favor low-cost solutions, MCMC methods balance exploration and exploitation. iSCO [42] introduces discrete LD [34, 35], leveraging gradient information and the MH [43] acceptance algorithm for parallelized neighborhood search. RLD [32] further incorporates a regularization term and simulated annealing (SA) to enhance exploration. Unlike classical Boltzmann Machine-based optimization [44], which relies on Gibbs sampling, RLD uses energy gradients to guide stochastic transitions toward promising regions. Recent collaborative neurodynamic optimization methods [45–47] also formulate COPs as QUBOs, but primarily rely on problem-specific deterministic dynamics. Building on RLD, NLD-EG further improves sampling efficiency by replacing random initialization with a learned warm-start distribution. More recently, FLD [48] replaces Gaussian noise with α-stable Lévy noise, enabling Lévy flights that facilitate escapes from deep, high-energy basins.

3 Preliminaries

Problem Definition. The COPs studied in this paper include MIS, MWIS, TSP, and CVRP. All these problems can be represented on a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{0, \cdots, N-1\}$ denotes the node set and $\mathcal{E}$ the edge set. Following previous works [5, 12], we encode a solution as a binary vector $\mathbf{x} \in \{0,1\}^M$. For MIS and MWIS, $M = N$ and $\mathbf{x}_i$ indicates whether node i is selected; for TSP and CVRP, $M = N^2$ and $\mathbf{x}_{i \cdot N + j}$ indicates whether edge (i, j) is selected. Mathematically, the goal is to find an optimal solution $\mathbf{x}^* = \underset{\mathbf{x} \in \Omega}{\operatorname{argmin}}\, c(\mathbf{x}, G)$, where Ω denotes the feasible set and $c(\cdot, \cdot)$ is the objective cost function. Formal definitions are provided in Appendix A.

Consistency Model. Consistency models [13, 38] define a family of mappings $f_\theta(\mathbf{x}_t, t)$ that transform a noisy sample $\mathbf{x}_t$ into a denoised version while preserving cross-time consistency. $\mathbf{x}_t$ is obtained by perturbing an initial sample $\mathbf{x}_0$ with noise at time step t. Under this formulation, a key property is that the time-step parameter t explicitly controls the noise level and thus the magnitude of the transformation. When t is small, $\mathbf{x}_t$ remains close to $\mathbf{x}_0$, leading to minor perturbations; larger t induces more significant deviations, enabling broader exploration of the solution space. This property makes consistency models well-suited for proposal generation in iterative search, providing a flexible and principled mechanism to balance exploitation and exploration.

Regularized Langevin Dynamics. RLD [32] is a stochastic search framework for discrete optimization that samples from a Boltzmann distribution $p(\mathbf{x}) \propto \exp(-H(\mathbf{x})/\tau)$ defined by an energy function $H(\mathbf{x})$. Starting from a solution $\mathbf{x}$, RLD constructs a proposal distribution $q(\mathbf{x}' \mid \mathbf{x})$ that combines gradient information with stochastic perturbations. To control the update magnitude, it imposes a regularization constraint on the expected step size: $\mathbb{E}_{q(\mathbf{x}'|\mathbf{x})}\left[\mathcal{D}(\mathbf{x}', \mathbf{x})\right] = \delta$, where $\mathcal{D}(\cdot, \cdot)$ denotes a distance metric and δ governs the exploration scale. Given a proposal $\mathbf{x}' \sim q(\mathbf{x}' \mid \mathbf{x})$, it is accepted under an SA criterion, favoring lower-energy states while allowing occasional uphill moves. After multiple iterations, a greedy decoding step is finally applied to ensure feasibility.

4 Methodology

4.1 Learning with Consistency Model

Diffusion Noising Process. Following the definitions in Sec. 3, the solution space is modeled by an M-dimensional Bernoulli distribution $p(\mathbf{x}) \in [0, 1]^{M\times 2}$. The goal of generative modeling is to learn the distribution of high-quality solutions conditioned on G, i.e., $p_\theta(\mathbf{x} \mid G)$. Given an initial solution $\mathbf{x}_0$, we follow prior diffusion models [5, 6, 13, 49] to define the noising process as below, where $\overline{\mathbf{Q}}_t = \mathbf{Q}_1 \cdots \mathbf{Q}_t$ is the cumulative transition matrix, and $\beta_t \in [0, 1]$ ensures doubly stochasticity.

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathbf{x}_0\overline{\mathbf{Q}}_t, \quad \mathbf{Q}_t = \begin{bmatrix} \beta_t & 1-\beta_t \\ 1-\beta_t & \beta_t \end{bmatrix}, \tag{1}$$

Consistency Learning. It is typically cast as supervised learning with a reference solution $\mathbf{x}^*$, often obtained from classical solvers for a given instance G. The discrete diffusion noising process defines a trajectory $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$ with $\mathbf{x}_0 = \mathbf{x}^*$ and $\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0)$. Following Fast-T2T [13], we use consistency learning to enable one-step inference, avoiding iterative modeling of $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, G)$ in vanilla diffusion. Specifically, we parameterize a consistency function $f_\theta(\mathbf{x}_t, t, G)$ that maps any noised point $\mathbf{x}_t$ to the optimal solution $\mathbf{x}^*$ conditioned on G. The training objective enforces consistency along the same trajectory:

$$\mathcal{L}(\theta) = \mathbb{E}\Big[d\big(f_\theta(\mathbf{x}_{t_1}, t_1, G), f_\theta(\mathbf{x}_{t_2}, t_2, G)\big)\Big]. \tag{2}$$

Similarly, we incorporate supervision from $\mathbf{x}^*$ by directly minimizing $d\big(f_\theta(\mathbf{x}_t, t, G), \mathbf{x}^*\big)$, which upper bounds the consistency objective via the triangle inequality:

$$\mathcal{L}(\theta) \leq \mathbb{E}\Big[d\big(f_\theta(\mathbf{x}_{t_1}, t_1, G), \mathbf{x}^*\big) + d\big(f_\theta(\mathbf{x}_{t_2}, t_2, G), \mathbf{x}^*\big)\Big], \tag{3}$$

Graph Neural Architectures. Prior works [5, 13] on consistency models typically parameterize the consistency function f_θ with Graph Convolutional Networks (GCNs) due to their simplicity and effectiveness in modeling local structures. However, GCNs exhibit notable limitations: they incur high memory overhead during training, hindering scalability to large batches, and their receptive field is tightly coupled with network depth, restricting the modeling of long-range dependencies without deep architectures. To address these issues, we additionally adopt Graph Attention Networks (GATs) as an alternative backbone. By leveraging attention mechanisms, GATs enable more flexible and adaptive information aggregation, thereby mitigating the limitations of GCNs. In this paper, we implement both backbones, with architectural details provided in Appendix D.

Data Augmentation. To address the data-intensive nature of consistency learning and mitigate the cost of acquiring high-quality supervisory labels for COPs, we design domain-specific data augmentation strategies to improve sample efficiency. For TSP and CVRP, we first perform coordinate normalization via translation and scaling. Specifically, we center TSP instances by shifting nodes

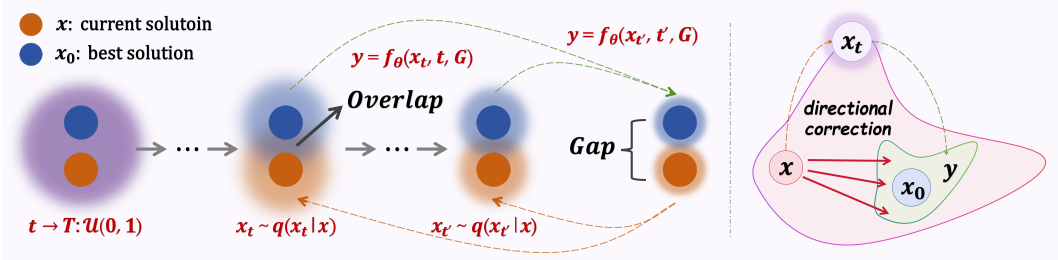

Figure 2: Illustration of distributional convergence and directional correction. $\mathcal{U}$: Uniform distribution, T: the total number of noising steps, we follow previous works [6, 12, 13] to set it as 1000.

to a zero-mean distribution, while for CVRP, we translate the depot to the origin. In both cases, coordinates are scaled by the mean ℓ_2 distance to ensure magnitude consistency. We then apply random geometric transformations, including rotations and horizontal, vertical, or central reflections. For MIS and MWIS, we exploit permutation invariance by randomly shuffling node indices and applying the corresponding permutations to the labels.

4.2 Solving with NLD Framework

Our proposed NLD framework is instantiated in two complementary variants: NLD-EG and NLD-IG, as described in Algorithm 1 and Algorithm 2, respectively.

NLD-EG: Neural-augmented RLD. For COPs with explicit energy formulations, we incorporate learned priors into RLD by replacing uninformed initialization with a data-driven alternative. Specifically, the model outputs a probability heatmap $p_\theta(\mathbf{x} \mid G)$ that characterizes the likelihood of each decision variable, preserving uncertainty while capturing global structural regularities. A natural approach is to directly decode the heatmap into a single deterministic solution, but this tends to concentrate on local optima and limit exploration. Therefore, we treat the heatmap as a factorized distribution and perform Bernoulli sampling to construct an initial set of solutions: $\mathbf{x}_0^{(k)} \sim p_\theta(\mathbf{x} \mid G)$. RLD is typically executed with k parallel chains, and this initialization provides a structured warm start that biases the search toward favorable regions. Each chain then evolves for c iterations, combining gradient-informed updates with step-size regularization controlled by δ and an acceptance mechanism governed by an initial temperature τ_0, to balance exploration and convergence.

NLD-IG: Consistency-driven Proposal Generation. For general COPs that are not readily cast into Ising or QUBO, we reformulate discrete search by lifting solutions into a continuous latent space via consistency models (CMs) [13, 38], which are designed to map any noisy state x_t directly back to the trajectory's origin x^* in a single step. This motivates us to leverage CMs to help escape local optima by denoising current solutions. However, there is a gap between the training stage and inference stage. While CMs are trained to recover the optimal structure x^* from $x_t \sim q(x \mid x^*)$, the current solution is typically suboptimal, meaning x_t is generated from $q(x_t \mid x)$, inducing a distributional discrepancy. Fortunately, this gap is a monotonically decreasing function of the noise level t. As theoretically validated in Fig. 2, as $t \to T$, both distributions converge to the same uniform prior, causing the gap to vanish. Despite this theoretical guarantee, practical CMs suffer from signal degradation at high noise levels, yielding overly diffuse predictions with limited utility. To navigate this trade-off, we adopt an annealing strategy for t during iterative search. Initially, we employ a high t to minimize the distributional gap, as the distance between the current solution x and x^* is unknown. As the search progresses and the solution quality improves, we gradually anneal t down to a lower value to incorporate stronger priors from the current solution, thereby enhancing the model's predictive fidelity. In each search iteration, given a noised state x_t constructed via categorical noise, we obtain the model prediction $f_\theta(x_t, t, G)$. Crucially, rather than directly adopting this prediction, we treat it as a directional correction along the consistency path and perform a controlled interpolation:

$$\mathbf{x}' = g\big(\mathbf{x} + \alpha \cdot (f_\theta(\mathbf{x}_t, t, G) - \mathbf{x})\big), \tag{4}$$

where $g(\cdot)$ is the decoding function, $\alpha \in (0.5, 1]$ controls the update strength and t determines the noise level of $\mathbf{x}_t$. Smaller α induces conservative refinements, while larger values encourage more aggressive transitions toward the model prediction. Empirically, $\alpha = 0.5$ serves as a transition threshold separating negligible from effective state changes under discrete decoding.

Algorithm 1 NLD-EG.

Input: Model $f_\theta(\cdot,\cdot,\cdot)$, problem instance G, energy function $H(\cdot)$, RLD hyperparameter (τ_0, δ, k, c), decoding function $g(\cdot,\cdot)$.
Initialize: $T \leftarrow 1000, \mathbf{x}_T \leftarrow \text{Bern}(p=0.5)$
Inference: $\mathcal{H}_\theta \leftarrow f_\theta(\mathbf{x}_T, T, G)$
Parallel k:
Sampling: $\mathbf{x} \sim \text{Bern}(\mathcal{H}_\theta),\ \mathbf{x}^* \leftarrow \mathbf{x}$
for $n = 1$ to c **do**
  $\tau \leftarrow \tau_0[1-(n-1)/c]$
  $\Delta \leftarrow (2\mathbf{x}-1) \odot \nabla H(\mathbf{x})$
  for $i = 1$ to N **do**
    $p \leftarrow \text{Sigmoid}\left((\Delta_i - \Delta_{(\delta)})/(2\tau)\right)$
    $u \sim \text{Bern}(p)$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i \cdot (1-u) + (1-\mathbf{x}_i) \cdot u$
  end for
  if $H(\mathbf{x}) < H(\mathbf{x}^*)$ **then**
    $\mathbf{x}^* \leftarrow \mathbf{x}$
  end if
end for
Decoding: $\mathbf{x}^* \leftarrow g(\mathbf{x}^*, G)$
Output: solution $\mathbf{x}^*$

Algorithm 2 NLD-IG.

Input: Model $f_\theta(\cdot,\cdot,\cdot)$, problem instance G, sequence of diffusion timesteps $1000 = \tau_0 > \tau_1 > \cdots > \tau_{N_\tau} = 0$ for solution initialization, $t_0 \geq t_1 \geq \cdots \geq t_{N_t-1}$ for exploration, sequence of alphas $\alpha_1 > \alpha_2 > \cdots > \alpha_{N_\alpha}$, decoding function $g(\cdot,\cdot)$, cost function $c(\cdot,\cdot)$.
Sample $\mathbf{x}_T$ from uniform distribution $\mathcal{U}$
Inference: $p_\theta(\mathbf{x}_0|G) \leftarrow f_\theta(\mathbf{x}_T, T, G)$
for $n = 1$ to $N_\tau - 1$ **do**
  $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|G),\ \mathbf{x}_{\tau_n} \sim \mathbf{x}_0 \overline{\mathbf{Q}}_{\tau_n}$
  $p_\theta(\mathbf{x}_0|G) \leftarrow f_\theta(\mathbf{x}_{\tau_n}, \tau_n, G)$
end for
Decoding: $\mathbf{x}^* \leftarrow g\big(p_\theta(\mathbf{x}_0|G); G\big)$
for $n = 0$ to $N_t - 1$ **do**
  $\mathbf{x}_{t_n} \sim \mathbf{x}^* \overline{\mathbf{Q}}_{t_n},\ \mathbf{y} \leftarrow f_\theta(\mathbf{x}_{t_n}, t_n, G)$
  for $m = 1$ to N_α **do**
    $\mathbf{x}_m \leftarrow g\big(\mathbf{x}^* + \alpha_m \cdot (\mathbf{y} - \mathbf{x}^*); G\big)$
  end for
  $\mathbf{x}^* \leftarrow \underset{x \in \mathcal{S}}{\text{argmin}}\, c(x, G),\ \mathcal{S} = \{\mathbf{x}^*, \mathbf{x}_1, \ldots, \mathbf{x}_{N_\alpha}\}$
end for
Output: Solution $\mathbf{x}^*$

This formulation enables multiple candidate solutions to be generated from a single model output by varying α, effectively amortizing computation. In contrast to classical MCMC or LD methods that rely on accept or reject dynamics, our approach performs deterministic, model-guided refinement while preserving an iterative search procedure. Theoretically, the update direction $\Delta = f_\theta(x_t, t, G) - x$ can be interpreted as an approximation to the implicit score function $\nabla \log p_\theta(x \mid G)$, following the framework of score-based generative models [38]. The consistency training objective enforces alignment of predictions across noise levels, thereby inducing a smooth vector field that points toward high-probability regions. As a result, the update rule can be viewed as a discretized gradient flow step over the learned probability landscape, where varying α further enables controllable exploration along this direction. Consequently, unlike classical LD that depends on an explicit energy gradient $\nabla H(x)$ and stochastic sampling, NLD-IG leverages a learned implicit gradient derived from the data distribution to perform efficient and directed search. The decoded solution $\mathbf{x}'$ can be refined via lightweight local search to further enhance feasibility and objective quality.

Decoding Functions and Post-processing Techniques. We map continuous solutions to the discrete domain via problem-specific decoding functions $g(\cdot)$, followed by lightweight post-processing. This design naturally bridges L2S and L2C, as discussed in Sec. 1. The model enables global exploration in continuous space, while decoding and post-processing ensure feasibility and improve solution quality in the discrete domain. Following recent L2C benchmarks [7, 50], we adopt greedy decoding, combined with a lightweight 2-OPT for TSP and Classic-LS [51] for CVRP. Such post-processing is standard practice in NCO, as greedy decoding alone often suffers from *prediction conflicts* [5]. Incorporating post-processing helps resolve these conflicts and better exploit the structural priors captured by the neural model. More detailed discussions are provided in Appendix C.

5 Experiments

5.1 Datasets and Metrics

In-distribution Datasets. We largely follow the protocol of the ML4CO-Bench-101 [7], while additionally incorporating larger-scale instances from GLOP [25] and RLD [32], including CVRP-1K, CVRP-2K, and ER-[9K-11K]. We use uniform graphs for TSP and CVRP; RB [52] and ER [53] graphs for MIS and MWIS. For CVRP, demands are sampled from [1, 10], with capacities of 40, 50, 80, 100, 200, and 300 for CVRP-50 to CVRP-2K, respectively. For MIS, the ER edge probability is 0.15 for ER-[700-800] and ER-[1400-1600], and 0.02 for ER-[9K-11K]. For MWIS, we follow the same RB construction as MIS and uniformly sample node weights from [0, 1].

Table 2: Results on TSP.

METHOD	TSP-100			TSP-500			TSP-1K		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
Concorde [4]	7.76*	0.00%	0.24s	16.55*	0.00%	18.67s	23.12*	0.00%	84.41s
LKH (500) [59]	7.76	0.00%	0.18s	16.55	0.00%	1.85s	23.12	0.01%	4.64s
GA-EAX [58]	7.76	0.00%	1.86s	16.55	0.00%	1.86s	23.12	0.00%	17.54s
SymNCO + 2OPT [8]	7.84	1.09%	0.10s	–	–	–	–	–	–
GOAL + 2OPT [10]	7.86	1.37%	0.76s	17.05	3.03%	4.06s	23.81	2.97%	8.07s
GCN + 2OPT [15]	7.78	0.24%	0.01s	16.77	1.35%	0.06s	23.53	1.77%	0.23s
DIMES + MCTS [16]	7.90	1.82%	0.03s	17.78	7.44%	0.83s	25.03	8.28%	3.87s
Fast-T2T + 2OPT [13]	7.76	0.08%	0.03s	16.69	0.87%	0.10s	23.42	1.32%	0.33s
COExpander + 2OPT [5]	7.76	0.06%	0.10s	16.63	0.49%	0.24s	23.34	0.95%	0.70s
NLD-IG-GAT (fewer)	7.76	0.03%	0.03s	16.57	0.15%	0.13s	23.19	0.33%	0.19s
MCMC (more) [43]	8.00	3.17%	0.07s	17.05	3.03%	0.36s	23.94	3.56%	0.73s
NLD-IG-GAT (more)	**7.76**	**0.01%**	0.09s	**16.56**	**0.06%**	0.30s	**23.15**	**0.16%**	0.43s
Fast-T2T + 2OPT (×8) [13]	7.76	0.01%	0.12s	16.61	0.37%	0.59s	23.31	0.84%	2.47s
COExpander + 2OPT (×8) [5]	7.76	0.01%	0.33s	16.58	0.18%	1.27s	23.24	0.52%	5.10s
NLD-IG-GAT (fewer) (×8)	7.76	0.01%	0.08s	16.55	0.04%	0.23s	23.15	0.14%	0.55s
MCMC (more) (×8) [43]	7.88	1.54%	0.14s	16.92	2.28%	0.54s	23.81	2.97%	1.04s
NLD-IG-GAT (more) (×8)	**7.76**	**0.00%**	0.14s	**16.55**	**0.02%**	0.55s	**23.13**	**0.06%**	1.28s

METHOD	TSP-2K			TSP-5K			TSP-10K		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
LKH (500) [59]	32.44*	0.00%	7.14s	50.97*	0.00%	34.49s	71.76*	0.00%	332.76s
GCN + 2OPT [15]	33.03	1.83%	0.95s	51.93	1.89%	7.33s	73.02	1.76%	28.29s
Fast-T2T + 2OPT [13]	33.00	1.75%	1.10s	51.80	1.65%	7.70s	72.98	1.71%	28.31s
COExpander + 2OPT [5]	32.91	1.45%	1.45s	51.67	1.38%	8.39s	72.83	1.50%	33.49s
NLD-IG-GCN (fewer)	32.64	0.61%	0.83s	51.50	1.05%	4.71s	72.69	1.31%	19.06s
MCMC (more) [43]	33.67	3.79%	3.64s	53.71	5.38%	38.40s	86.90	21.10%	68.84s
NLD-IG-GCN (more)	**32.54**	**0.33%**	2.09s	**51.28**	**0.61%**	13.98s	**72.47**	**1.00%**	55.95s

Out-of-distribution (OOD) and Real-world Datasets. For OOD evaluation, we use Gaussian and Clustered distributions for TSP, and Holme-Kim (HK) [54] and Watts-Strogatz (WS) [55] graphs for MIS. For real-world evaluation, we use 49 TSP instances (51-1002 nodes) from TSPLIB[2], and 70 CVRP instances (101-513 nodes) from the X set of [56] in CVRPLIB[3].

Metrics. 1) Objective. The average objective of solutions w.r.t. the corresponding instances. 2) Drop. The relative performance drop w.r.t. the objective compared to the solutions obtained by the baseline exact solver or heuristic solver. 3) Time. The average computational time per instance.

5.2 Evaluated Methods and Model Settings

Compared solvers. 1) Baseline. We adopt Gurobi [3], Concorde [4], KaMIS [57], GA-EAX [58], HGS [51] and LKH [59]. Besides, we also include several learning-free algorithms as fast baselines, with implementation details in Appendix B. 2) Construction-based. We select several recent representative works from ML4CO-Bench-101 [7], including SymNCO [8], GCN [15], DIMES [16], UTSP [19], DiffUCO [14], Fast-T2T [13], COExpander [5], GOAL [10], Meta-EGN [60], LwD [21], VAG-CO [61]. 3) Search-based. We select NeuOpt-GIRE [29], MCMC [43] as a learning-free search baseline in Appendix B, and classical LD-based solvers, including iSCO [42], RLD [32] and FLD [48]. 4) D&C. We select GLOP [25], TAM-LKH [62], and UDC [26].

Training Settings. 1) GCN-based backbone. Following previous works [5, 6, 12, 13], we employ a 12-layer GCN architecture for all tasks. The feature dimension is set to 128 for MIS and MWIS, and 256 for TSP and CVRP. 2) GAT-based backbone. For MIS and MWIS, we adopt a 16-layer encoder where timestep embeddings are injected into node features at the input embedding layer. For TSP and CVRP, we utilize an encoder-decoder structure consisting of a 16-layer encoder and a 6-layer decoder. The feature dimension is set to 128 for all tasks. 3) For both backbones, the initial learning rate is set to 0.0002, and each dataset is trained for a total of 500,000 steps.

Solving Settings. 1) To ensure a unified end-to-end pipeline, we use the trained consistency model itself for initialization. Specifically, we perform 5 initialization steps (I_s) for TSP, and 1 step for the remaining. 2) NLD-EG. We follow the settings of RLD [32] and FLD [48], including the initial temperature τ_0, regularization term δ, penalty coefficient β, number of parallel chains k, and number

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

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

Table 3: Results on CVRP.

METHOD	CVRP-50			CVRP-100		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
HGS [51]	10.37*	0.00%	1.01s	15.56*	0.00%	20.03s
Fast-T2T + LS [13]	10.87	4.84%	0.01s	16.29	4.70%	0.02s
COExpander + LS [5]	10.77	3.90%	0.04s	16.22	4.25%	0.06s
GOAL + LS [10]	10.63	2.52%	0.51s	15.96	2.55%	0.97s
SymNCO + LS [8]	10.56	1.91%	0.09s	15.93	2.38%	0.17s
NLD-IG-GAT (fewer)	10.47	0.99%	0.03s	15.75	1.19%	0.04s
NeuOpt-GIRE (T=1k) [29]#	10.43	0.61%	0.08s	15.87	1.94%	0.17s
NLD-IG-GAT (more)	**10.39**	**0.24%**	0.10s	**15.63**	**0.41%**	0.18s
Fast-T2T + LS [13] (×8)	10.70	3.18%	0.04s	16.11	3.53%	0.10s
COExpander + LS [5] (×8)	10.57	1.98%	0.12s	15.95	2.46%	0.31s
NLD-IG-GAT (fewer) (×8)	10.40	0.34%	0.05s	15.65	0.53%	0.08s
NeuOpt-GIRE (T=5k) [29]#	10.38	0.16%	0.35s	15.70	0.87%	0.83s
NLD-IG-GAT (more) (×8)	**10.37**	**0.07%**	0.18s	**15.59**	**0.16%**	0.29s

METHOD	CVRP-200			CVRP-500		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
HGS [51]	19.63*	0.00%	60.02s	37.15*	0.00%	360.38s
Fast-T2T + LS [13]	20.66	5.29%	0.06s	39.20	5.53%	0.22s
COExpander + LS [5]	20.59	4.89%	0.15s	39.12	5.34%	0.61s
GOAL + LS [10]	20.40	3.94%	2.04s	38.12	2.62%	7.31s
SymNCO + LS [8]	20.19	2.88%	0.34s	38.70	4.17%	0.88s
NLD-IG-GAT (fewer)	19.91	1.43%	0.10s	37.96	2.18%	0.32s
MCMC (more) [43]	20.65	5.17%	0.72s	39.32	5.63%	3.41s
NLD-IG-GAT (more)	**19.75**	**0.60%**	0.42s	**37.66**	**1.36%**	1.43s
Fast-T2T + LS [13] (×8)	20.40	3.96%	0.31s	38.79	4.44%	1.61s
COExpander + LS [5] (×8)	20.27	3.26%	0.94s	38.65	4.04%	4.76s
NLD-IG-GAT (fewer) (×8)	19.77	0.73%	0.23s	37.78	1.68%	0.94s
MCMC (more) [43] (×8)	20.31	3.45%	0.96s	38.84	4.54%	4.19s
NLD-IG-GAT (more) (×8)	**19.69**	**0.28%**	0.87s	**37.49**	**0.92%**	3.85s

METHOD	CVRP-1K			CVRP-2K		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
HGS [51]	41.17*	0.00%	360.05s	57.18*	0.00%	360.30s
TAM-LKH [62]#	46.30	12.46%	1.80s	64.80	13.32%	5.60s
GLOP-LKH [25]#	45.90	11.49%	1.10s	63.00	10.18%	1.50s
UDC [26]#	43.48	5.61%	8.40s	60.94	6.57%	13.80s
NLD-IG-GAT (fewer)	42.85	4.12%	1.32s	59.43	4.02%	5.37s
MCMC (more) [43]	43.87	6.65%	6.76s	60.20	5.43%	31.52s
NLD-IG-GAT (more)	**42.30**	**2.78%**	6.21s	**58.33**	**2.09%**	24.84s

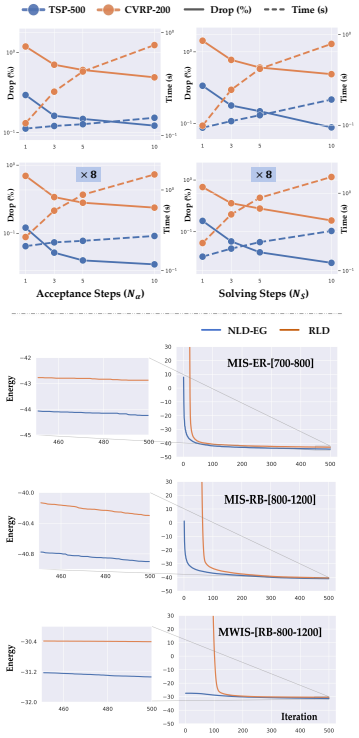

Figure 3: Ablation study on acceptance and solving steps; Energy landscape comparisons between RLD and NLD-EG.

of iterations c. Detailed configurations are provided in Table 6. 3) NLD-IG. This variant involves the number of acceptance steps N_α, the number of solving steps N_s, along with the noise time range. Detailed configurations are provided in Table 7. 4) Unless otherwise specified, all testing experiments are conducted with a batch size set to 1. In particular, for the TSP and CVRP, we follow prior works [5–7, 33] to additionally report results with 8 parallel runs.

Note on the Implementation. 1) For RLD [32], we reproduce the method using its official open-source implementation and conduct a fresh evaluation. 2) For iSCO [42] and FLD [48], whose code is unavailable, we mark them with # and report the results from FLD, the more recent work. 3) For NeuOpt-GIRE [29], TAM-LKH [62], GLOP [25], and UDC [26], also marked with #, we use results reported in their original papers. Their runtimes were typically measured with large batch sizes, which can underestimate the actual cost of instance-by-instance inference. For example, NeuOpt-GIRE (T=1K) reports 0.08s on CVRP-50 under batched evaluation, compared to 32.49s in our sequential setting. To remain consistent with prior work and since this paper is not a benchmarking study, we retain the published runtimes. Even under this baseline-favorable protocol, our method remains highly competitive. A systematic comparison under unified parallel inference settings is left for future work. 4) Notably, several works on CVRP use LKH [59] as the baseline, and we instead adopt the stronger HGS [51], yielding substantially improved reference performance.

5.3 Experimental Results

Overview. To ensure fair performance comparisons, we equip NLD4CO with two configuration sets, *fewer* and *more*. Their detailed settings are given in Table 6 and Table 7, corresponding to the typical time budgets of construction-based and search-based solvers, respectively. The main results for the four COPs considered in this paper are presented in Table 2 (TSP), Table 3 (CVRP), and Table 4 (MIS and MWIS). More comprehensive results are provided in Appendix F, including Table 13 (MWIS), Table 14 (MIS), Table 15 (TSP), and Table 16 (CVRP). ***A unified comparison with both construction-based and search-based solvers under fair settings is summarized in Table 19.***

Table 4: Results on MIS and MWIS.

METHOD	MIS-RB-[200-300]			MIS-RB-[800-1200]			MIS-RB-[2K-3K]		
	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓
KaMIS [57]	20.09*	0.00%	45.81s	43.00*	0.00%	56.97s	49.26*	0.00%	60.96s
iSCO [42]#	19.29	3.98%	0.33s	36.96	14.05%	1.35s	–	–	–
FLD [48]#	20.02	0.35%	0.08s	40.25	6.40%	0.23s	–	–	–
RLD (SA) [32]	19.93	0.75%	0.08s	40.56	5.57%	0.14s	46.38	5.71%	0.38s
NLD-EG-GAT (fewer)	20.02	0.36%	0.09s	41.18	4.16%	0.17s	46.90	4.68%	0.42s
GCN [15]	18.17	9.41%	0.01s	36.06	16.04%	0.05s	38.76	21.22%	0.24s
VAG-CO [61]	17.31	13.71%	0.02s	34.00	20.78%	0.03s	37.26	24.20%	0.15s
Meta-EGN [60]	17.69	11.78%	0.08s	34.67	19.26%	0.13s	38.84	21.04%	0.74s
Fast-T2T [13]	19.33	3.71%	0.19s	38.94	9.39%	0.71s	38.38	21.99%	8.98s
COExpander [5]	19.66	2.09%	0.11s	41.23	4.06%	0.62s	45.84	6.86%	5.34s
NLD-EG-GAT (more)	**20.08**	**0.06%**	0.12s	**42.50**	**1.12%**	0.50s	**48.64**	**1.22%**	3.38s
METHOD	MIS-ER-[700-800]			MIS-ER-[1400-1600]			MIS-ER-[9K-11K]		
	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓
KaMIS [57]	44.97*	0.00%	60.75s	50.94*	0.00%	60.82s	381.44*	0.00%	623.37s
iSCO [42]#	42.18	6.20%	0.68s	–	–	–	365.37	4.21%	247.50s
FLD [48]#	44.37	1.33%	0.15s	–	–	–	377.50	1.03%	4.20s
RLD (SA) [32]	44.21	1.68%	0.15s	49.80	2.23%	0.28s	375.63	1.52%	4.52s
NLD-EG-GCN (fewer)	44.45	1.14%	0.17s	50.03	1.77%	0.35s	378.88	0.67%	5.03s
GCN [15]	35.85	20.27%	0.04s	37.68	26.02%	0.12s	296.00	22.39%	1.23s
VAG-CO [61]	33.81	24.80%	0.03s	36.91	27.53%	0.07s	287.38	24.66%	1.03s
Meta-EGN [60]	33.15	26.28%	0.12s	37.56	26.26%	0.27s	–	–	–
LwD [21]	37.29	17.07%	0.59s	39.52	22.40%	1.47s	–	–	–
Fast-T2T [13]	40.20	10.62%	0.64s	44.09	13.43%	5.12s	327.56	14.12%	27.55s
COExpander [5]	42.38	5.75%	0.47s	48.16	5.43%	2.21s	366.75	3.85%	24.64s
NLD-EG-GCN (more)	**45.08**	**-0.25%**	0.26s	**51.55**	**-1.21%**	2.11s	**383.94**	**-0.66%**	23.07s
METHOD	MWIS-RB-[200-300]			MWIS-RB-[800-1200]			MWIS-RB-[2K-3K]		
	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓
Gurobi [3]	14.86*	0.00%	0.19s	32.93*	0.00%	39.82s	37.97*	0.00%	53.80s
RLD (SA) [32]	14.70	1.05%	0.08s	30.36	7.44%	0.14s	35.12	7.27%	0.30s
NLD-EG-GCN (fewer)	14.82	0.26%	0.09s	31.50	3.97%	0.17s	36.46	3.39%	0.40s
GCN [15]	14.18	4.65%	0.01s	27.80	15.35%	0.04s	29.30	22.96%	0.11s
Fast-T2T [13]	14.44	2.85%	0.17s	27.94	14.91%	0.37s	30.52	19.54%	3.83s
NLD-EG-GCN (more)	**14.85**	**0.04%**	0.12s	**32.72**	**0.27%**	0.50s	**38.41**	**-1.49%**	3.10s

NLD-IG. 1) TSP. Under the *fewer* setting, it lowers the L2C average drop from 0.84% to 0.53%. Under the *more* setting, it markedly outperforms search-based methods, bringing the drop down from 6.18% to **0.34%** with a **94.6%** improvement. 2) CVRP. It improves upon L2C SOTA solvers by decreasing the drop from 3.66% to 1.90%. When compared with search-based solvers under the *more* setting, it further narrows the drop from 3.52% to **1.05%**, achieving a **70.2%** improvement.

NLD-EG. 1) MIS. Compared to L2C solvers under the *more* setting, NLD-EG narrows the average drop from 4.67% to **0.01%**, representing a **99.8%** improvement. For search-based methods in the *fewer* setting, NLD-EG reduces the drop from 2.70% to 2.12%. 2) MWIS. Under the *more* setting, NLD-EG achieves an average drop of **-0.44%** with a **25**× speedup over Gurobi. 3) Energy Landscape Trajectory. We illustrate the convergence behavior, as visualized in Fig. 3, NLD-EG exhibits a lower initial energy descent and rapidly stabilizes at a superior plateau compared to RLD.

5.4 Ablation Study and Generalization Study

Ablation Study on N_s and N_α. We first study the effect of N_α with $N_s = 5$ and observe that increasing N_α consistently improves performance. However, as shown in Fig. 3, the gains gradually saturate, exhibiting diminishing returns relative to the additional runtime cost. To balance efficiency and performance, we set $N_\alpha = 5$ in our final configuration. We then perform ablations on N_s with $N_\alpha = 5$. As expected, increasing N_s further improves performance. However, compared to acceptance steps, scaling N_s incurs higher computational overhead due to additional model inference, leading to increased runtime. Therefore, in practice, N_s should be chosen based on the desired trade-off between solution quality and computational budget.

Ablation Study on Backbones. We evaluate both GAT and GCN backbones (Appendix D) across all benchmark problems and observe: 1) For NLD-IG on routing tasks, GAT achieves faster inference

and slightly better performance on small- to medium-scale instances. However, as the scale increases, GAT becomes less efficient than GCN in both training and inference stages. This scaling trend is consistent with the higher computational cost of dense attention, whereas GCN benefits from efficient sparse message passing. 2) For NLD-EG on MIS and MWIS, GCN and GAT perform comparably, suggesting that the specific choice of backbone has limited impact in this setting and both neural architectures are sufficient to capture the underlying problem structure.

Ablation Study on Annealing and Warm-start. Ablations on the annealing schedule and warm-start strategy are provided in the Appendix F. For NLD-EG, linear temperature annealing is more robust than fixed temperatures under the *fewer* setting, avoiding excessive stochasticity at high temperatures and premature convergence at low temperatures (Table 9). For NLD-IG, annealing the diffusion timestep t outperforms fixed timesteps on TSP and CVRP under the *more* setting, as t controls the search neighborhood and progressively shrinks it from exploration to refinement (Table 10). For warm starts, RLD is robust to initialization quality but benefits from greater diversity. Identical LC-Degree solutions across parallel chains limit exploration, while bit-flip noise with ϵ mitigates this issue. The neural warm start provides a balance between quality and diversity through heatmap-based Bernoulli sampling, yielding the strongest post-RLD results on MIS (Table 11).

Generalization Study on Out-of-Distribution Datasets. For TSP, we evaluate on Gaussian and clustered distributions. As shown in Table 5 of Appendix C, NLD-IG demonstrates strong generalization across both cross-scale and cross-distribution settings. For MIS, we evaluate on HK and WS graphs, with results presented in Table 12 of Appendix F.

Generalization Study on Real-world Datasets. We evaluate generalization on TSPLIB and CVRPLIB, with results reported in Tables 17 and 18. NLD-IG generalizes well and substantially outperforms the previous L2C SOTA, COExpander [5]. It reduces the average optimality gaps from 0.36% and 5.47% to **0.09%** and **0.98%**, yielding relative reductions of **75.0%** and **82.1%**, respectively.

6 Extended Discussion and Conclusion

Energy-Landscape. From an energy-landscape perspective, NLD-EG and NLD-IG alleviate the local-optima challenge in complementary ways. ***NLD-EG improves where the search begins.*** Instead of random initialization, it provides diverse initial states in relatively low-energy regions learned from high-quality solutions, allowing RLD to reach promising basins earlier and devote more iterations to effective refinement. In contrast, ***NLD-IG improves how the search moves.*** Classical LD and MCMC-based COP solvers typically rely on local proposals, such as 2-OPT for routing, which restrict transitions to the neighborhood of the current solution. Moving between low-energy basins often requires crossing higher-energy regions, where MH acceptance is low, particularly at low temperatures, while subsequent local transitions may return the search to the original basin. NLD-IG instead constructs a learned transition pathway beyond the local neighborhood. Controllable noising moves the current solution away from its basin while preserving structural information, and denoising then guides it toward another potentially promising region. The annealing schedule further balances basin exploration and local refinement by controlling the perturbation level throughout the search.

Generative Backbone Agnosticism. Although NLD-IG is instantiated with a consistency model [38], its design is not intrinsically tied to it. The proposal generator only needs to predict a refined solution conditioned on the current state, providing a data-driven optimization direction. Consistency models thus represent one effective instantiation rather than a fundamental component of NLD-IG. Other few-step generative models can be incorporated under the same principle, including emerging flow- and drift-based models [63, 64] for conditional proposal generation. Exploring stronger generative backbones and their interaction with NLD remains an interesting direction for future work.

Conclusion. We propose NLD4CO, a unified framework that integrates LD with data-driven learning for COPs. By introducing two complementary instantiations, NLD-EG and NLD-IG, the framework systematically addresses the core limitations of classical LD. Specifically, NLD-EG enhances sampling efficiency in energy-based settings through neural-guided warm-start initialization, effectively reducing mixing time while preserving the original gradient-driven dynamics. In contrast, NLD-IG generalizes LD beyond explicit energy formulations by leveraging learned implicit gradients derived from consistency models, enabling structured and globally coordinated search in complex discrete spaces. Extensive experiments across diverse COPs demonstrate that NLD4CO achieves strong performance in terms of both solution quality and computational efficiency.

References

[1] 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.

[2] 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.

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

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

[5] 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.

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

[7] J. Ma, W. Pan, Y. Li, and J. Yan, "Ml4co-bench-101: Benchmark machine learning for classic combinatorial problems on graphs," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[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] 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.

[17] 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.

[18] 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.

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

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

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

[22] W. Pan, S. Yan, N. Chen, Y. Qu, J. Ma, and J. Yan, "UECO: A unified encoder for neural combinatorial optimization via iterative edge-node reasoning," in *The Fortieth Annual Conference on Neural Information Processing Systems*, 2026.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] P. R. d O Costa, J. Rhuggenaath, Y. Zhang, and A. Akcay, "Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning," in *Asian conference on machine learning*. PMLR, 2020, pp. 465–480.

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

[32] S. Feng and Y. Yang, "Regularized langevin dynamics for combinatorial optimization," *arXiv preprint arXiv:2502.00277*, 2025.

[33] 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 *Advances in Neural Information Processing Systems*, 2025.

[34] R. Zhang, X. Liu, and Q. Liu, "A langevin-like sampler for discrete distributions," in *International Conference on Machine Learning*. PMLR, 2022, pp. 26 375–26 396.

[35] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, p. 823, 1930.

[36] E. Ising, "Beitrag zur theorie des ferromagnetismus," *Zeitschrift für Physik*, vol. 31, no. 1, pp. 253–258, 1925.

[37] A. Lucas, "Ising formulations of many np problems," *Frontiers in physics*, vol. 2, p. 5, 2014.

[38] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *International Conference on Machine Learning*, 2023.

[39] 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.

[40] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[41] Y. Ba, X. Lin, C. Zhou, R. Zheng, Z. Wang, X. Liang, Z. Lu, J. Sun, Y. Qian, and Q. Zhang, "Survey on neural routing solvers," *arXiv preprint arXiv:2602.21761*, 2026.

[42] 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.

[43] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[44] J. H. Korst and E. H. Aarts, "Combinatorial optimization on a boltzmann machine," *Journal of Parallel and distributed computing*, vol. 6, no. 2, pp. 331–357, 1989.

[45] H. Li and J. Wang, "A collaborative neurodynamic algorithm for quadratic unconstrained binary optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 9, no. 1, pp. 228–239, 2024.

[46] H. Li, J. Wang, N. Zhang, and W. Zhang, "Binary matrix factorization via collaborative neurodynamic optimization," *Neural Networks*, vol. 176, p. 106348, 2024.

[47] H. Li and J. Wang, "Capacitated clustering via majorization-minimization and collaborative neurodynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 5, pp. 6679–6692, 2022.

[48] 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.

[49] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg, "Structured denoising diffusion models in discrete state-spaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 981–17 993, 2021.

[50] 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.

[51] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Operations Research*, vol. 60, no. 3, pp. 611–624, 2012.

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

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

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

[55] D. J. Watts and S. H. Strogatz, "Collective dynamics of âĂŸsmall-worldâĂŹnetworks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[56] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 257, no. 3, pp. 845–858, 2017.

[57] 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.

[58] Y. Nagata and S. Kobayashi, "A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem," *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 346–363, 2013.

[59] 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.

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

[61] 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.

[62] 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.

[63] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022.

[64] M. Deng, H. Li, T. Li, Y. Du, and K. He, "Generative modeling via drifting," *arXiv preprint arXiv:2602.04770*, 2026.

[65] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne *et al.*, "Jax: composable transformations of python+ numpy programs," 2018.

[66] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[67] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in neural information processing systems*, vol. 32, 2019.

Appendix

A Problem Definitions

Maximum Independent Set (MIS). Given G, an independent set $\mathcal{S} \subseteq \mathcal{V}$ is a subset of nodes such that no two nodes in $\mathcal{S}$ are adjacent. The goal is to maximize $|\mathcal{S}|$ s.t. $\forall i, j \in \mathcal{S}$, $(i, j) \notin \mathcal{E}$.

Maximum Weight Independent Set (MWIS). Given G, each node $i \in \mathcal{V}$ is associated with a weight $w_i \in \mathbb{R}_+$. The goal is to find an independent set that maximizes the total weight $\sum_{i \in \mathcal{S}} w_i$, subject to $\forall i, j \in \mathcal{S}$, $(i, j) \notin \mathcal{E}$.

Traveling Salesman Problem (TSP). Given G and a cost matrix $\mathcal{C} \in \mathbb{R}^{N \times N}$, where $\mathcal{C}_{ij}$ denotes the cost of edge (i, j), the objective is to find a tour $\tau = (i_1, \cdots, i_N)$ that minimizes the total cost: $\sum_{k=1}^{N-1} \mathcal{C}_{i_k i_{k+1}} + \mathcal{C}_{i_N i_1}$.

Capacitated Vehicle Routing Problem (CVRP). Given G, a depot node $v_0 \in \mathcal{V}$, a cost matrix $\mathcal{C} \in \mathbb{R}^{N \times N}$, a demand vector $\mathcal{D} \in \mathbb{R}_+^N$, and a vehicle capacity $\mathcal{Q} > 0$, the goal is to plan a set of routes $\mathcal{R}$, each route $r \in \mathcal{R}$ starting and ending at the depot v_0, such that each customer node is visited exactly once and the total demand on each route does not exceed $\mathcal{Q}$, i.e., $\sum_{i \in r} \mathcal{D}_i \leq \mathcal{Q}$. The objective is to minimize the total cost of all routes: $\min_{\mathcal{R}} \sum_{r \in \mathcal{R}} \sum_{(i,j) \in r} \mathcal{C}_{ij}$.

Energy functions for MIS and MWIS. Following previous works [14, 61, 37], we define the energy functions for MIS and MWIS, where β is a penalty coefficient for constraint violation. For MIS, the energy function is defined as $H(\mathbf{x}) = -\sum_{i=1}^{N} x_i + \beta \sum_{(i,j) \in \mathcal{E}} x_i x_j$. For MWIS, the energy function is defined as $H(\mathbf{x}) = -\sum_{i=1}^{N} w_i x_i + \beta \sum_{(i,j) \in \mathcal{E}} x_i x_j$, where w denotes node weights.

B Learning-free Baselines

Recent years have seen rapid progress in combinatorial optimization, particularly in NCO, where a wide range of learning-based solvers have been proposed. Despite these advances, there is still a lack of systematic efforts to establish simple yet essential learning-free baselines that can serve as standardized reference points for pipelines such as L2C and L2S. To bridge this gap, we develop a suite of learning-free baseline solvers for the four classes of combinatorial optimization problems considered in this paper. For each problem, we design tailored methods that exploit its structural properties, spanning both construction-based heuristics and MCMC-based search algorithms. All solvers are implemented in C to ensure high computational efficiency. This section aims to provide a unified and fair evaluation framework for learning-based methods, enabling a clearer quantification of the performance gains achieved by neural approaches over simple yet efficient baselines.

Learning-free Construction-based Baselines. We first consider a set of classical construction-based heuristics that build feasible solutions in a sequential manner. These methods typically start from an empty solution and iteratively extend it by selecting the next element according to problem-specific greedy criteria. For routing problems, we adopt insertion and nearest-neighbor methods for TSP (Algorithm 3 and Algorithm 6) and a nearest-neighbor heuristic for CVRP (Algorithm 4), which incrementally construct tours by minimizing marginal costs or local distances. For graph problems, we employ degree-based greedy strategies for MIS and MWIS (Algorithm 5 and Algorithm 7), which iteratively select low (weighted) degree nodes while maintaining feasibility via masking.

Learning-free Search-based Baselines.

We further include a search-based baseline that refines solutions via stochastic exploration, as shown in Algorithm 8. Specifically, we adopt a general MCMC [43] framework with reversible proposal operators and a temperature-controlled acceptance criterion. Note that we restrict the proposals to be reversible, i.e., $q(x' \mid x) = q(x \mid x')$. This makes the Hastings correction term equal to one

$$\frac{q(x \mid x')}{q(x' \mid x)} = 1, \tag{5}$$

so that the acceptance probability reduces to $\alpha = \min\{1, \exp(-\Delta/\tau)\}$. Finally, a decoding function is applied to map the final state to a feasible solution, ensuring constraint satisfaction.

Algorithm 3 Insertion for TSP

Input: Cost matrix $\mathcal{C}$, random permutation except the first node $\{p_1, \ldots, p_{n-1}\}$.
Initialize: $\mathcal{S} \leftarrow \{v_1, v_1\}$
for $i = 1$ to $n-1$ **do**
  $v \leftarrow v_{p_i}$
  $(a, b) \leftarrow \underset{(a,b)\in\mathcal{S}}{\operatorname{argmin}} \mathcal{C}_{a,v} + \mathcal{C}_{v,b} - \mathcal{C}_{a,b}$
  Insert v between a and b in $\mathcal{S}$
end for
Output: solution $\mathcal{S}$

Algorithm 4 Nearest for CVRP

Input: Cost matrix $\mathcal{C}$, normalized demands $\mathcal{D}$.
Initialize: $\mathcal{S} \leftarrow \{v_0\}, \mathcal{U} \leftarrow \mathcal{V} \setminus \{v_0\}$
Initialize: $v \leftarrow v_0, r \leftarrow 1.0$
while $\mathcal{U} \neq \varnothing$ **do**
  $\mathcal{Y} \leftarrow \{y \in \mathcal{U} \mid \mathcal{D}_y \leq r\}$
  if $\mathcal{Y} = \varnothing$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{v_0\}, v \leftarrow v_0, r \leftarrow 1.0$
  else
    $y \leftarrow \underset{y\in\mathcal{Y}}{\operatorname{argmin}} \mathcal{C}_{v,y}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{y\}, \mathcal{U} \leftarrow \mathcal{U} \setminus \{y\}$
    $r \leftarrow r - \mathcal{D}_y, v \leftarrow y$
  end if
end while
$\mathcal{S} \leftarrow \mathcal{S} \cup \{v_0\}$
Output: solution $\mathcal{S}$

Algorithm 5 GP-Degree for MIS and MWIS

Input: Adjacency matrix $\mathcal{A}$, weights w.
Initialize: $\mathcal{S} \leftarrow \{0\}^n, \mathcal{M} \leftarrow \{0\}^n$
Get Degrees: $\mathcal{D}_i \leftarrow -w_i + \sum_{j=1}^n w_j \cdot \mathcal{A}_{i,j}$
Sort nodes in an ascending order according to $\mathcal{D}$: $\{v_{k_1}, v_{k_2}, \cdots, v_{k_n}\}$
for $i = 1$ to n **do**
  $k \leftarrow k_i$
  if $\mathcal{M}_k \neq 1$ **then**
    $\mathcal{M}_j \leftarrow 1, \forall j \in \mathcal{J} = \{j \mid \mathcal{A}_{k,j} = 1\}$
    $\mathcal{M}_k \leftarrow 1, \mathcal{S}_k \leftarrow 1$
  end if
end for
Output: solution $\mathcal{S}$

Algorithm 6 Nearest for TSP

Input: Cost matrix $\mathcal{C}$.
Initialize: $\mathcal{S} \leftarrow \{v_1\}, \mathcal{U} \leftarrow \mathcal{V} \setminus \{v_1\}, v \leftarrow v_1$
for $i = 1$ to $n-1$ **do**
  $u \leftarrow \underset{u\in\mathcal{U}}{\operatorname{argmin}} \mathcal{C}_{v,u}$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{u\}, \mathcal{U} \leftarrow \mathcal{U} \setminus \{u\}, v \leftarrow u$
end for
$\mathcal{S} \leftarrow \mathcal{S} \cup \{v_1\}$
Output: solution $\mathcal{S}$

Algorithm 7 LC-Degree for MIS and MWIS

Input: Adjacency matrix $\mathcal{A}$, weights w.
Initialize: $\mathcal{S} \leftarrow \{0\}^n, \mathcal{M} \leftarrow \{0\}^n$
for $i = 1$ to n **do**
  $i \leftarrow \underset{\{i|\mathcal{M}_i\neq 1\}}{\operatorname{argmin}} -w_i + \sum_{j=1}^n w_j \cdot \mathcal{A}_{i,j}$
  $\mathcal{J} = \{j \mid \mathcal{A}_{ij} = 1\}$
  $\mathcal{M}_j \leftarrow 1, \mathcal{S}_j \leftarrow 0, \forall j \in \mathcal{J}$
  $\mathcal{A}_{j,\cdot} \leftarrow \{0\}^N, \mathcal{A}_{\cdot,j} \leftarrow \{0\}^N, \forall j \in \mathcal{J}$
  $\mathcal{M}_i \leftarrow 1, \mathcal{S}_i \leftarrow 1$
end for
Output: solution $\mathcal{S}$

Algorithm 8 MCMC

Input: Problem instance G, initial solution $\mathcal{S}_0$, temperature sequence $\{\tau_t\}_{t=1}^T$, reversible proposal operators $\mathcal{Q} = \{q_m\}_{m=1}^M$, energy function $E(\cdot)$, decoding function $g(\cdot)$.
Initialize: $x \leftarrow \mathcal{S}_0, x^* \leftarrow \mathcal{S}_0$.
Initialize: $\mathcal{H} \leftarrow E(x, G), \mathcal{H}^* \leftarrow E(x^*, G)$.
for $t = 1$ to T **do**
  Proposal: $q \sim \mathcal{Q}, x' \sim q(\cdot \mid x)$.
  Energy difference: $\Delta_t \leftarrow E(x') - E(x)$.
  $\alpha_t \leftarrow \min\{1, \exp(-\Delta_t/\tau_t)\}, u \sim \mathcal{U}(0, 1)$.
  if $u \leq \alpha_t$ **then**
    $x \leftarrow x', \mathcal{H} \leftarrow \mathcal{H} + \Delta_t$.
    if $\mathcal{H} < \mathcal{H}^*$ **then**
      $x^* \leftarrow x,\ \mathcal{H}^* \leftarrow \mathcal{H}$.
    end if
  end if
end for
$\mathcal{S} \leftarrow g(x^*)$
Output: solution $\mathcal{S}$

C Further Discussion on the Pipelines for NCO

C.1 Learning to Construct

For this pipeline, we try to revisit a central question in the NCO literature: ***to what extent is data-driven (neural) learning genuinely effective, and does it encounter intrinsic scalability bottlenecks as problem size grows?*** We will examine this question through three representative COPs.

On TSP. We adopt GCN [15] and COExpander [5] as representative L2C methods. Several key observations emerge from Fig. 4. 1) Heuristics such as Insertion and Nearest exhibit remarkable stability across scales, with performance largely insensitive to problem size. Notably, Nearest combined with 2-OPT consistently maintains an average optimality gap of approximately 5.0%, even comparable to the best results reported in prior works (e.g., GLOP [25], DIMES [16], and Att-GCN [17]). 2) Without 2-OPT, L2C methods degrade rapidly as problem size increases, falling behind Insertion on large instances (e.g., 5K and 10K). 3) Most interestingly, neural methods demonstrate substantial improvement when equipped with 2-OPT. We attribute this to their high edge prediction accuracy: in our experiments, learning-based models achieve nearly 90.0% correctness in edge selection even on TSP-10K. However, a purely greedy decoding process fails to fully exploit this advantage, as *prediction conflicts* [5] tend to accumulate in later stages. The application of 2-OPT effectively mitigates such conflicts, thereby unlocking the latent potential of neural approaches. Furthermore, the fact that Nearest outperforms Insertion also underscores the importance of edge accuracy, since greedy edge selection is more likely to capture correct edges.

On CVRP. We adopt SymNCO [8], GOAL [10], and COExpander [5] as representative methods. As shown in Fig. 4, these approaches exhibit highly consistent performance trends, and the introduction of local search does not alter their relative rankings. Owing to the intrinsic characteristics of the CVRP, the baseline heuristics are comparatively weaker, which allows learning-based methods to consistently demonstrate stronger performance. This also highlights the effectiveness of machine learning approaches for problems like CVRP.

On MIS. We compare GCN [15] and COExpander [5] as representative L2C methods with the learning-free baselines, including GP-Degree and LC-Degree. As MIS does not commonly employ standard local search procedures, all methods are directly evaluated using greedy decoding on both RB and ER graphs. As shown in Fig. 4, two main observations can be drawn: 1) GCN achieves performance comparable to GP-Degree but remains inferior to LC-Degree, which is consistent with our observations during training. This can be attributed to the intrinsic properties of MIS (such as the existence of multiple optimal solutions and large gaps between near-optimal ones), which make the learning process less effective and lead to overly smooth, averaged predictions. 2) COExpander [5] delivers significantly stronger performance, benefiting both from its consistency-based training strategy and its solving paradigm (i.e., Adaptive Expansion). In particular, its progressive reduction of problem size is well aligned with MIS, where instances can be naturally decomposed into smaller subproblems, thereby improving optimization effectiveness.

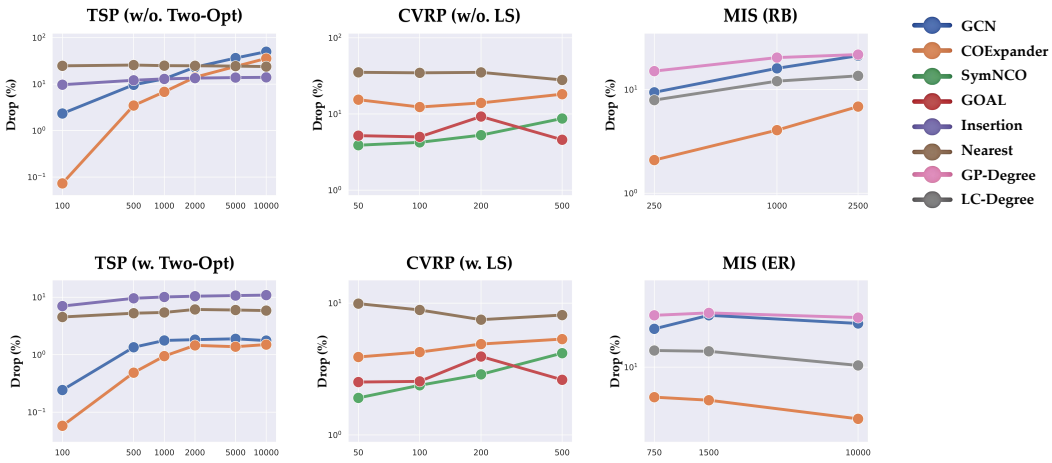

Figure 4: Performance comparison of L2C methods and learning-free baselines across scales.

C.2 Learning to Search

For the pipeline of L2S, we focus on the performance, efficiency, and generalization of GenSCO [33] on TSP. Admittedly, GenSCO achieves remarkable success, improving both solution quality and efficiency by nearly an order of magnitude. However, these gains largely hinge on two key factors: the adoption of rectified flow models and the use of modern high-performance frameworks such as JAX [65] and Flax[4], with the latter contributing substantially to computational speedups.

Under a comparable GAT-based architecture, a single round of model inference combined with decoding and 2-OPT reduces the total solving time on TSP-500 from 15.499s to 3.052s. At the same time, the rectified flow formulation significantly improves solution quality, reducing the optimality gap from 0.577% to 0.038%. These gains primarily stem from advances in model design and system-level optimization, rather than from changes in the underlying solving framework. For the sake of fair comparison and reproducibility, our implementation is based on PyTorch and NumPy without leveraging such acceleration techniques, allowing us to more clearly isolate the contributions of modeling and algorithmic components from engineering optimizations.

As for the rectified flow models, they may exhibit limitations in generalization, as indicated by recent studies [41]. We further validate this phenomenon through empirical analysis. In particular, we conduct comprehensive generalization evaluations across three distributions (Uniform, Gaussian, and clustered) and four problem scales (100, 200, 500, and 1K), covering both cross-scale and cross-distribution settings. The Gaussian and clustered instances are adopted from ML4CO-Bench-101 [7], and the results are summarized in Table 5.

It can be observed that GenSCO exhibits clear limitations in generalization, suffering from significant performance degradation under both scale and distribution shifts. In contrast, our method maintains consistently stable performance across all settings.

Table 5: Generalization Performance across Scales and Distributions for TSP.

Method	Model	TSP-100		TSP-500		TSP-1K		Average	
		Obj.↓	Drop↓	Obj.↓	Drop↓	Obj.↓	Drop↓	Obj.↓	Drop↓
GenSCO (C=20) (×8) [33]	TSP-100	7.756	0.000%	19.000	14.832%	31.400	35.825%	**19.39**	**16.89%**
GenSCO (C=20) (×8) [33]	TSP-500	18.713	141.281%	16.549	0.017%	23.147	0.125%	19.47	47.14%
GenSCO (C=20) (×8) [33]	TSP-1K	34.479	344.554%	16.550	0.028%	23.131	0.054%	24.72	114.88%
NLD-IG-GAT (more) (×8)	TSP-100	7.756	0.001±0.008%	16.643	0.587±0.172%	23.381	1.136±0.197%	15.93	0.58%
NLD-IG-GAT (more) (×8)	TSP-500	7.762	0.077±0.139%	16.549	0.018±0.029%	23.174	0.244±0.116%	15.83	0.11%
NLD-IG-GAT (more) (×8)	TSP-1K	7.766	0.126±0.195%	16.552	0.039±0.049%	23.132	0.058±0.044%	**15.82**	**0.07%**
NLD-IG-GCN (more) (×8)	TSP-100	7.756	0.003±0.019%	16.598	0.313±0.156%	23.337	0.945±0.186%	15.90	0.42%
NLD-IG-GCN (more) (×8)	TSP-500	7.758	0.022±0.072%	16.552	0.040±0.047%	23.147	0.127±0.127%	**15.82**	**0.06%**
NLD-IG-GCN (more) (×8)	TSP-1K	7.760	0.058±0.136%	16.552	0.036±0.044%	23.142	0.102±0.058%	15.82	0.07%
Method	Model	Gaussian-100		Gaussian-200		Gaussian-500		Average	
		Obj.↓	Drop↓	Obj.↓	Drop↓	Obj.↓	Drop↓	Obj.↓	Drop↓
GenSCO (C=20) (×8) [33]	TSP-100	34.031	0.001%	48.163	0.075%	89.911	15.983%	**57.37**	**5.35%**
GenSCO (C=20) (×8) [33]	TSP-500	63.271	85.922%	143.431	198.028%	515.957	565.575%	240.89	283.18%
GenSCO (C=20) (×8) [33]	TSP-1K	74.280	118.271%	180.202	274.435%	584.478	653.965%	279.65	348.89%
NLD-IG-GAT (more) (×8)	TSP-100	34.033	0.006±0.044%	48.142	0.032±0.075%	78.001	0.620±0.205%	53.39	0.22%
NLD-IG-GAT (more) (×8)	TSP-500	34.074	0.126±0.173%	48.147	0.042±0.093%	77.600	0.101±0.100%	**53.27**	**0.09%**
NLD-IG-GAT (more) (×8)	TSP-1K	34.093	0.182±0.211%	48.214	0.180±0.161%	77.971	0.580±0.218%	53.43	0.31%
NLD-IG-GCN (more) (×8)	TSP-100	34.040	0.026±0.090%	48.172	0.094±0.111%	77.893	0.480±0.173%	53.37	0.20%
NLD-IG-GCN (more) (×8)	TSP-500	34.046	0.044±0.118%	48.142	0.033±0.076%	77.584	0.082±0.078%	**53.26**	**0.05%**
NLD-IG-GCN (more) (×8)	TSP-1K	34.061	0.087±0.163%	48.143	0.035±0.063%	77.575	0.070±0.069%	53.26	0.06%
Method	Model	Cluster-100		Cluster-200		Cluster-500		Average	
		Obj.↓	Drop↓	Obj.↓	Drop↓	Obj.↓	Drop↓	Obj.↓	Drop↓
GenSCO (C=20) (×8) [33]	TSP-100	5.528	0.025%	6.920	0.115%	13.335	24.364%	**8.59**	**8.17%**
GenSCO (C=20) (×8) [33]	TSP-500	8.649	54.497%	7.165	3.648%	10.795	0.676%	8.87	19.61%
GenSCO (C=20) (×8) [33]	TSP-1K	11.032	99.626%	9.433	36.462%	10.881	1.478%	10.45	45.86%
NLD-IG-GAT (more) (×8)	TSP-100	5.526	0.002±0.016%	6.913	0.010±0.029%	10.749	0.246±0.119%	7.73	0.09%
NLD-IG-GAT (more) (×8)	TSP-500	5.528	0.024±0.072%	6.914	0.024±0.052%	10.733	0.095±0.107%	**7.73**	**0.05%**
NLD-IG-GAT (more) (×8)	TSP-1K	5.528	0.026±0.071%	6.914	0.029±0.060%	10.738	0.139±0.124%	7.73	0.07%
NLD-IG-GCN (more) (×8)	TSP-100	5.528	0.028±0.134%	6.915	0.036±0.078%	10.753	0.276±0.166%	7.73	0.11%
NLD-IG-GCN (more) (×8)	TSP-500	5.532	0.097±0.253%	6.916	0.050±0.106%	10.742	0.174±0.150%	**7.73**	**0.11%**
NLD-IG-GCN (more) (×8)	TSP-1K	5.532	0.110±0.249%	6.917	0.066±0.114%	10.745	0.208±0.200%	7.73	0.13%

[4] https://github.com/google/flax

D Graph Neural Architecture

Previous works [5, 6, 12, 13] have predominantly adopted GCNs as the backbone architecture, owing to their effectiveness in capturing local graph structures and scalability to large instances. In this paper, we go beyond this standard choice by considering both GCN and GAT backbones, enabling a systematic investigation of their respective strengths, limitations, and applicability across different problem settings. The following presents the detailed architecture used in our framework.

D.1 Graph Convolutional Networks

Embedding Layers. Given node features ($x \in \mathbb{R}^{|\mathcal{V}|\times 2}$) and edge features ($e \in \mathbb{R}^{|\mathcal{E}|}$), we follow [6, 12, 15] and compute sinusoidal features for each input element, which serve as positional encodings for nodes ($\widetilde{x}$), edges ($\widetilde{e}$), and timesteps ($\widetilde{t}$), respectively. Specifically, $\widetilde{x}_i = \mathrm{concat}(\widetilde{x}_{i,0}, \widetilde{x}_{i,1})$, and

$$\widetilde{x}_{i,j} = \mathrm{concat}\left(\sin\frac{x_{i,j}}{T^{\frac{0}{d}}}, \cos\frac{x_{i,j}}{T^{\frac{0}{d}}}, \sin\frac{x_{i,j}}{T^{\frac{2}{d}}}, \cos\frac{x_{i,j}}{T^{\frac{2}{d}}}, \ldots, \sin\frac{x_{i,j}}{T^{\frac{d-2}{d}}}, \cos\frac{x_{i,j}}{T^{\frac{d-2}{d}}}\right), \tag{6}$$

$$\widetilde{e}_i = \mathrm{concat}\left(\sin\frac{e_i}{T^{\frac{0}{d}}}, \cos\frac{e_i}{T^{\frac{0}{d}}}, \sin\frac{e_i}{T^{\frac{2}{d}}}, \cos\frac{e_i}{T^{\frac{2}{d}}}, \ldots, \sin\frac{e_i}{T^{\frac{d-2}{d}}}, \cos\frac{e_i}{T^{\frac{d-2}{d}}}\right), \tag{7}$$

$$\widetilde{t} = \mathrm{concat}\left(\sin\frac{t}{T^{\frac{0}{d}}}, \cos\frac{t}{T^{\frac{0}{d}}}, \sin\frac{t}{T^{\frac{2}{d}}}, \cos\frac{t}{T^{\frac{2}{d}}}, \ldots, \sin\frac{t}{T^{\frac{d-2}{d}}}, \cos\frac{t}{T^{\frac{d-2}{d}}}\right), \tag{8}$$

where d denotes the embedding dimension, T is set to 10000 by default, and $\mathrm{concat}(\cdot)$ denotes concatenation. The denoising timestep t, required by the consistency model, satisfies $t \in \{\tau_1, \ldots, \tau_M\}$. Note that $\widetilde{x}_{i,j}$ and $\widetilde{e}_i$ denote the corresponding entries of the embedded vectors $\widetilde{x}$ and $\widetilde{e}$, respectively. The resulting input features to the graph convolutional network are then projected via linear transformations with learnable parameters $W_1^0, W_2^0, W_3^0, W_4^0$:

$$\mathbf{h}_v^0 = W_1^0\widetilde{x}, \quad \mathbf{h}_e^0 = W_2^0\widetilde{e}, \quad \mathbf{h}_t^0 = W_4^0(\mathrm{ReLU}(W_3^0\widetilde{t})). \tag{9}$$

Graph Convolutional Layers. Let $h_{v,i}^l$ and $h_{e,ij}^l$ denote the hidden representations of node i and edge (i,j) at layer l, respectively. The cross-layer convolution operation is defined as

$$h_{v,i}^{l+1} = h_{v,i}^l + \mathrm{ReLU}\Big(\mathrm{BN}\big(W_1^l h_{v,i}^l + \sum_{j\sim i}\eta_{ij}^l \odot W_2^l h_{v,j}^l\big)\Big), \tag{10}$$

$$h_{e,ij}^{l+1} = h_{e,ij}^l + \mathrm{ReLU}\big(\mathrm{BN}\big(W_3^l h_{e,ij}^l + W_4^l h_{v,i}^l + W_5^l h_{v,j}^l\big)\big), \tag{11}$$

$$\eta_{ij}^l = \frac{\sigma(h_{e,ij}^l)}{\sum_{j'\sim i}\sigma(h_{e,ij'}^l) + \epsilon}, \tag{12}$$

where $W_1, \cdots, W_5 \in \mathbb{R}^{h\times h}$ are model parameters, and η_{ij}^l denotes the dense attention map. This operation incorporates edge representations to capture the importance of edges in routing problems.

For TSP, we fuse the timestep feature with the edge convolutional feature and reformulate its update:

$$h_{e,ij}^{l+1} = h_{e,ij}^l + \mathrm{ReLU}\big(\mathrm{BN}\big(W_3^l h_{e,ij}^l + W_4^l h_{v,i}^l + W_5^l h_{v,j}^l\big)\big) + W_6^l\big(\mathrm{ReLU}\big(\mathbf{h}_t^0\big)\big). \tag{13}$$

For MIS, likewise, the timestep feature is fused with the node convolutional feature:

$$h_{v,i}^{l+1} = h_{v,i}^l + \mathrm{ReLU}\Big(\mathrm{BN}\big(W_1^l h_{v,i}^l + \sum_{j\sim i}\eta_{ij}^l \odot W_2^l h_{v,j}^l\big)\Big) + W_6^l\big(\mathrm{ReLU}\big(\mathbf{h}_t^0\big)\big). \tag{14}$$

Output Layer. The edge heatmap for TSP and the node heatmap for MIS are predicted as

$$p_{e,ij} = \mathrm{Softmax}\Big(\mathrm{norm}\left(\mathrm{ReLU}(W_e h_{e,ij}^L)\right)\Big), \quad p_{v,i} = \mathrm{Softmax}\big(\mathrm{norm}\big(\mathrm{ReLU}\big(W_n h_{v,i}^L\big)\big)\big) \tag{15}$$

respectively, where L is the number of GCN layers and norm denotes layer normalization.

D.2 Graph Attention Networks

Embedding Layers. The embedding procedure for GAT mirrors that of GCN, i.e., sinusoidal positional encoding followed by linear projection, but excludes edge embeddings, as the multi-head attention mechanism implicitly models pairwise interactions over the complete graph.

Graph Attention Encoder. The encoder consists of L_{enc} stacked pre-norm Transformer layers operating on the node embeddings $\mathbf{h}_v^l \in \mathbb{R}^h$. Each layer applies multi-head self-attention (MHSA) followed by a SwiGLU feed-forward network (FFN), both with residual connections:

$$\mathbf{h}_v^{l+\frac{1}{2}} = \mathbf{h}_v^l + \mathrm{MHSA}\big(\mathrm{LN}\big(\mathbf{h}_v^l\big)\big), \quad \mathbf{h}_v^{l+1} = \mathbf{h}_v^{l+\frac{1}{2}} + \mathrm{SwiGLU}\Big(\mathrm{LN}\big(\mathbf{h}_v^{l+\frac{1}{2}}\big)\Big), \tag{16}$$

where LN denotes layer normalization. Within each MHSA, queries, keys, and values are obtained via a joint projection $W^{qkv} \in \mathbb{R}^{3h\times h}$, and the attention scores are stabilized by a softcap operation:

$$A_{ij} = \delta_2 \cdot \tanh\left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\delta\sqrt{d_k}}\right), \qquad \mathbf{o}_i = \sum_j \mathrm{Softmax}_j(A_{ij})\,\mathbf{v}_j, \tag{17}$$

where $\delta_2 = 30.0$ is the softcap constant and $d_k = h/N_{\text{heads}}$ is the per-head dimension. The encoder output $\mathbf{h}_v^{L_{\text{enc}}}$ is further projected by a linear layer $W_{\text{out}} \in \mathbb{R}^{h\times h}$.

Graph Attention Decoder. The decoder consists of L_{dec} pre-norm Transformer layers that condition the encoded node features on the noised solution matrix $s_t \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and the timestep embedding $\mathbf{h}_t^0$. Each decoder layer pairs a GAT layer with a Feature-wise Linear Modulation (FiLM) layer [66], which maps the timestep embedding to affine parameters (γ^l, β^l) via a two-layer MLP:

$$\begin{bmatrix}\gamma^l \\ \beta^l\end{bmatrix} = \tanh\Big(W_2^{f,l}\Big(\mathrm{SiLU}\Big(W_1^{f,l}\,\mathbf{h}_t^0\Big)\Big)\Big), \tag{18}$$

where $W_1^{f,l} \in \mathbb{R}^{h\times h}$, $W_2^{f,l} \in \mathbb{R}^{2\times h}$, and the two scalar outputs γ^l, β^l are broadcast over s_t. The modulated solution matrix then serves as an additive attention bias in MHSA, so that the noised solution directly biases which node-pairs receive high attention, providing condition information across all decoder layers:

$$s_t^l = (1+\gamma^l)\,s_t + \beta^l, \quad A_{ij}^l = \delta \cdot \tanh\left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\delta\sqrt{d_k}}\right) + s_{tij}^l. \tag{19}$$

Output Layer. Afterwards, an MLP with SiLU activation and RMSNorm [67] projects each node embedding to its final representation $\mathbf{z}_v \in \mathbb{R}^h$. The edge heatmap is then obtained via bilinear scoring $p_{ij} = \mathrm{Softmax}_j\big(\mathbf{z}_i^\top \mathbf{z}_j\big)$, where $\mathrm{Softmax}$ is applied row-wise over all candidate neighbors of node i.

D.3 Comparative Analysis of GCN and GAT

From the experimental results, we observe that: 1) GCNs benefit from their simple and efficient aggregation scheme, which makes them well-suited for large-scale problems and stable training; however, their uniform (or weakly adaptive) message passing may limit their ability to capture fine-grained, heterogeneous interactions. 2) In contrast, GATs introduce attention mechanisms that allow for adaptive weighting of neighboring nodes, offering greater expressiveness and the potential to model more complex relational patterns. This flexibility, however, comes at the cost of increased computational overhead and may lead to reduced scalability or training instability on large graphs.

E Broader Impacts, Future Work, Limitations and Clarification

Broader Impacts. This work advances learning-based approaches for COPs by proposing NLD4CO, a unified framework that integrates data-driven models with classical LD. Our methods have the potential to improve solution quality and efficiency in real-world applications, including logistics, transportation, and resource allocation. Besides, we provide a set of learning-free baselines, which enable a clearer and more rigorous assessment of the actual gains brought by learning-based methods.

Future Work. We plan to extend NLD4CO to a broader range of COPs, particularly more challenging routing variants such as ATSP and CVRPTW. Additionally, we will investigate the integration of the D&C strategy to further scale NLD4CO to larger-scale COPs.

Limitations. Despite the strong empirical performance, several limitations remain.

- NLD4CO is inherently constrained by the scalability limitations of NCO. Extending to very large instances (e.g., beyond 10K nodes) remains challenging due to dataset availability, training complexity, and memory constraints, which are common issues across NCO methods.
- NLD-IG relies on the L2C pipeline when mapping from the continuous space back to the discrete domain. Therefore, it requires that the underlying problem admits an effective solution within the L2C framework, including a fast decoding function and lightweight local search operators.
- The applicability of NLD-IG depends on certain structural properties of the problem. Particularly, it assumes that high-quality solutions lie in a relatively smooth solution manifold, where local refinements can effectively improve solution quality. This assumption holds for routing problems such as TSP and CVRP, where good solutions often share similar structures, but may not hold for problems like MIS, where optimal and near-optimal solutions can differ significantly. As a result, the model-guided interpolation may fail to provide meaningful improvements.

Clarification about Contributions. We clarify that NLD-EG is not intended to be the primary contribution of this work. We include NLD-EG within the NLD4CO framework for two reasons: 1) NLD-EG ensures the completeness of the framework by retaining the strengths of LD-based methods on QUBO problems. 2) We observe that several mainstream LD-based methods, such as iSCO [42] and FLD [48], do not provide publicly available implementations, while the released code of RLD [32] exhibits certain issues in practice. We therefore correct these issues and conduct experiments based on the fixed implementation to ensure a fair comparison. Besides, to support future research in the community, we further implement the MCMC baselines for each problem considered in this work. We emphasize that our main contribution lies in integrating data-driven modeling with LD, and in proposing a more general perspective of "gradient-guided proposals" as a unified search paradigm, which extends LD methods to a broader class of COPs.

F Supplementary Experiments and Details

Hardware. All models are trained and tested using NVIDIA H800 (80G) GPUs and an Intel(R) Xeon(R) Platinum 8558 96-Core Processor CPU. For all test evaluations, a single GPU is utilized, and the batch size is set to 1 to ensure a fair comparison of the solving time across different models.

Compared Solver Settings. We follow the settings in ML4CO-Bench-101 [7]. For Fast-T2T [13], we use inference steps $I_s = 20$ for MIS and MWIS, $I_s = 5$ for TSP, and $I_s = 1$ for CVRP. For COExpander [5], we use determination steps $D_s = 20$ and $I_s = 1$ for MIS, $D_s = 3$ and $I_s = 5$ for TSP, and $D_s = 3$ and $I_s = 1$ for CVRP.

Training and Solving Settings. 1) Our training data is sourced from ML4CO-Bench-101 [7]. 2) To ensure a fair comparison with existing construction-based and search-based methods, we equip each solver with two sets of configurations, denoted as *fewer* and *more*, as detailed in Table 6 and Table 7.

Complete Experiments. We provide complete experimental results, covering not only our method but also baselines, construction-based methods, search-based methods, and representative divide-and-conquer approaches. Specifically, the results include: ablation study for NLD-IG (Table 8), generalization study for MIS (Table 12), complete results on MWIS (Table 13), MIS (Table 14), TSP (Table 15), and CVRP (Table 16), along with evaluations on TSPLIB (Table 17) and CVRPLIB (Table 18), comparisons with prior SOTA methods (Table 19).

Table 6: Solving Settings for NLD-EG and MCMC. τ_0: initial temperature, δ: regularization term, β: penalty coefficient, k: parallel chains, c: number of iterations, f: fewer, m: more.

DATASET	PRETRAINED	SETTINGS (NLD-EG)							SETTINGS (MCMC)			
		τ_0	δ	β	k^f	c^f	k^m	c^m	τ_0	τ_{end}	N_s^f	N_s^m
MIS-RB-[200-300]	MIS-RB-[200-300]	0.01	5	1.02	200	300	1000	500	1.0	0.001	1e6	2e6
MIS-RB-[800-1200]	MIS-RB-[800-1200]	0.01	5	1.02	200	500	1000	2000	1.0	0.001	1e6	3e6
MIS-RB-[2K-3K]	MIS-RB-[800-1200]	0.01	5	1.02	200	1000	1000	5000	1.0	0.001	3e6	2e7
MIS-ER-[700-800]	MIS-ER-[700-800]	0.01	20	1.001	200	500	1000	1000	1.0	0.001	1e6	2e6
MIS-ER-[1400-1600]	MIS-ER-[700-800]	0.01	20	1.001	200	1000	1000	5000	1.0	0.001	2e6	1e7
MIS-ER-[9K-11K]	MIS-ER-[700-800]	0.01	20	1.001	200	5000	500	10000	1.0	0.001	5e7	1e8
MWIS-RB-[200-300]	MWIS-RB-[200-300]	0.01	5	1.02	200	300	1000	500	1.0	0.001	1e6	2e6
MWIS-RB-[800-1200]	MWIS-RB-[800-1200]	0.01	5	1.02	200	500	1000	2000	1.0	0.001	1e6	5e6
MWIS-RB-[2K-3K]	MWIS-RB-[800-1200]	0.01	5	1.02	200	1000	1000	5000	1.0	0.001	2e6	2e7

Table 7: Solving Settings for NLD-IG and MCMC, f: fewer, m: more.

DATASET	PRETRAINED	SETTINGS (NLD-IG)					SETTINGS (MCMC)			
		t	I_s	N_α	N_s^f	N_s^m	τ_0	τ_{end}	N_s^f	N_s^m
TSP-100	TSP-100	(250, 750)	5	5	1	5	0.01	0.001	1e6	2e6
TSP-500	TSP-500	(250, 750)	5	5	5	20	0.01	0.001	5e6	1e7
TSP-1K	TSP-1K	(250, 750)	5	5	5	20	0.01	0.001	1e7	2e7
TSP-2K	TSP-1K	(250, 750)	5	5	5	20	0.01	0.001	5e7	1e8
TSP-5K	TSP-10K	(250, 750)	5	5	5	20	0.01	0.001	2e8	5e8
TSP-10K	TSP-10K	(250, 750)	5	5	5	20	0.01	0.001	2e8×3	2e8×9
CVRP-50	CVRP-50	(250, 500)	1	5	1	5	0.01	0.001	–	–
CVRP-100	CVRP-100	(250, 500)	1	5	1	5	0.01	0.001	–	–
CVRP-200	CVRP-200	(250, 500)	1	5	1	5	0.01	0.001	–	1e6
CVRP-500	CVRP-500	(250, 500)	1	5	1	5	0.01	0.001	–	2e6
CVRP-1K	CVRP-500	(250, 500)	1	5	1	5	0.01	0.001	–	2e6
CVRP-2K	CVRP-500	(250, 500)	1	5	1	5	0.01	0.001	–	5e6

Table 8: Ablation study on solving steps and acceptance steps for NLD-IG-GAT.

SETTINGS			TSP-500		CVRP-200		SETTINGS			TSP-500		CVRP-200	
N_s	N_α	Parallel	DROP↓	TIME↓	DROP↓	TIME↓	N_s	N_α	Parallel	DROP↓	TIME↓	DROP↓	TIME↓
5	1	1	0.294%	0.117s	1.186%	0.132s	1	5	1	0.336%	0.094s	1.431%	0.099s
5	3	1	0.162%	0.123s	0.703%	0.267s	3	5	1	0.178%	0.111s	0.772%	0.246s
5	5	1	0.148%	0.129s	0.604%	0.419s	5	5	1	0.148%	0.129s	0.604%	0.419s
5	10	1	0.122%	0.149s	0.489%	0.757s	10	5	1	0.088%	0.192s	0.488%	0.788s
5	1	8	0.121%	0.201s	0.696%	0.261s	1	5	8	0.160%	0.152s	0.731%	0.228s
5	3	8	0.052%	0.224s	0.340%	0.554s	3	5	8	0.065%	0.193s	0.356%	0.533s
5	5	8	0.040%	0.234s	0.281%	0.873s	5	5	8	0.040%	0.234s	0.281%	0.873s
5	10	8	0.035%	0.269s	0.238%	1.557s	10	5	8	0.025%	0.326s	0.165%	1.620s

Table 9: Ablation on temperature annealing for NLD-EG (GCN, *fewer*).

DATASET	Fixed temperature					Annealing (linear decrease)				
	1.0	0.5	0.2	0.1	0.01	1.0	0.5	0.2	0.1	0.01
MIS-RB-[200-300]	7.227%	6.934%	2.590%	0.497%	0.386%	2.156%	1.323%	0.534%	0.442%	0.288%
MIS-RB-[800-1200]	16.604%	16.550%	12.126%	6.587%	4.761%	9.528%	7.856%	6.129%	5.353%	4.246%
MIS-ER-[700-800]	21.412%	17.852%	3.016%	1.227%	1.419%	4.735%	2.497%	1.383%	1.261%	1.139%
MIS-ER-[1400-1600]	22.032%	19.392%	2.748%	1.598%	1.936%	3.822%	2.517%	1.600%	1.691%	1.768%

Table 10: Ablation on diffusion-timestep annealing for NLD-IG (GAT, *more*).

DATASET	Fixed timestep				ANNEALING
	1000	750	500	250	
TSP-500	0.148%	0.076%	0.069%	0.076%	0.062%
TSP-1K	0.191%	0.186%	0.172%	0.217%	0.156%
CVRP-200	0.663%	0.645%	0.615%	0.735%	0.604%
CVRP-500	1.457%	1.400%	1.379%	1.387%	1.364%

Table 11: Ablation on warm-start strategies for NLD-EG (GCN, *fewer*). We report the drop before and after RLD. Random independently sets each variable to 0 or 1 with equal probability. Insertion draws a random variable order and sequentially adds a variable to the independent set if doing so does not violate independence constraints. For all methods, feasibility is guaranteed by sequential decoding. For LC-Degree, ϵ denotes the bit-flip probability used to inject initialization diversity.

WARM-STARTS	MIS-RB-[200-300]		MIS-RB-[800-1200]		MIS-ER-[700-800]		MIS-ER-[1400-1600]	
	w/o. RLD	w. RLD	w/o. RLD	w. RLD	w/o. RLD	w. RLD	w/o. RLD	w. RLD
Random	10.419%	0.750%	17.447%	5.573%	22.596%	1.682%	24.348%	2.227%
Insertion	10.013%	0.564%	18.407%	4.849%	22.581%	2.078%	24.427%	2.476%
GP-Degree	15.277%	1.616%	20.635%	4.991%	25.867%	1.472%	26.942%	1.982%
LC-Degree ($\epsilon = 0$)	7.916%	2.139%	12.058%	4.933%	13.595%	2.741%	13.395%	2.571%
LC-Degree ($\epsilon = 0.05$)	–	1.489%	–	4.590%	–	1.921%	–	2.046%
LC-Degree ($\epsilon = 0.1$)	–	1.191%	–	4.569%	–	1.782%	–	1.971%
LC-Degree ($\epsilon = 0.2$)	–	0.745%	–	5.094%	–	1.522%	–	2.166%
NLD-EG-GCN	5.546%	0.288%	15.360%	4.246%	18.825%	1.139%	22.554%	1.768%

Table 12: Generalization Study on MIS. NLD-EG adopts the pre-trained model of ER-[700-800].

METHOD	SETTINGS					MIS-HK-[200-300]			MIS-WS-[200-300]		
	τ_0	δ	β	k	c	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓
KaMIS [57]	–	–	–	–	–	79.372*	0.000±0.000%	54.174s	76.904*	0.000±0.000%	51.490s
RLD [32]	0.01	5	1.001	1000	500	79.370	0.003±0.057%	0.147s	76.902	0.003±0.064%	0.146s
NLD-EG-GAT	0.01	5	1.001	1000	500	79.370	0.002±0.050%	0.160s	76.904	0.000±0.000%	0.160s
NLD-EG-GCN	0.01	5	1.001	1000	500	79.370	0.002±0.050%	0.163s	76.904	0.000±0.000%	0.162s
METHOD	SETTINGS					MIS-HK-[800-1200]			MIS-WS-[800-1200]		
	τ_0	δ	β	k	c	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓
KaMIS [57]	–	–	–	–	–	330.946*	0.000±0.000%	67.272s	262.570*	0.000±0.000%	37.792s
RLD [32]	0.01	5	1.001	1000	1000	330.608	0.010±0.152%	0.382s	259.718	1.069±0.385%	0.348s
NLD-EG-GAT	0.01	5	1.001	1000	1000	330.772	0.051±0.114%	0.395s	260.126	0.743±0.336%	0.352s
NLD-EG-GCN	0.01	5	1.001	1000	1000	330.626	0.092±0.147%	0.408s	260.578	0.745±0.343%	0.363s

Table 13: Complete results on MWIS.

METHOD	MWIS-RB-[200-300]			MWIS-RB-[800-1200]			MWIS-RB-[2K-3K]		
	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓
Baseline Solvers									
Gurobi [3]	14.856*	0.000%	0.192s	32.927*	0.000%	39.819s	37.971*	0.000%	53.798s
KaMIS [57]	14.636	1.513%	10.058s	28.030	14.613%	61.061s	34.259	9.200%	60.150s
GP-Degree	11.562	22.416%	0.000s	22.521	31.478%	0.004s	23.937	37.182%	0.022s
LC-Degree	13.316	10.494%	0.000s	27.481	16.366%	0.008s	30.060	20.917%	0.057s
Construction-based Solvers with Greedy Decoding									
GCN [15]	14.177	4.653%	0.010s	27.795	15.350%	0.042s	29.300	22.960%	0.107s
Fast-T2T [13]	14.439	2.851%	0.166s	27.943	14.909%	0.371s	30.522	19.537%	3.827s
Search-based Solvers									
RLD (SA) [32]	14.698	1.053%	0.078s	30.361	7.440%	0.137s	35.124	7.274%	0.295s
RLD (SA) (more) [32]	14.801	0.367%	0.117s	31.508	4.012%	0.478s	36.587	3.392%	3.082s
MCMC (fewer)	14.054	5.423±3.188%	0.088s	29.660	9.608±10.164%	0.151s	33.952	10.382±4.511%	0.548s
MCMC (more)	14.132	4.901±3.044%	0.174s	30.551	6.907±9.602%	0.729s	35.245	6.961±4.407%	5.074s
NLD-EG-GAT (fewer)	14.811	0.301±0.543%	0.092s	31.372	4.354±9.761%	0.158s	36.412	3.758±6.024%	0.410s
NLD-EG-GAT (more)	14.847	0.057±0.198%	0.118s	32.597	0.632±10.201%	0.484s	38.464	-1.623±5.281%	3.117s
NLD-EG-GCN (fewer)	14.818	0.262±0.535%	0.094s	31.504	3.970±10.048%	0.169s	36.457	3.617±6.536%	0.398s
NLD-EG-GCN (more)	14.847	0.043±0.150%	0.118s	32.717	0.272±10.347%	0.496s	38.413	-1.489±5.508%	3.100s

Table 14: Complete results on MIS.

METHOD	RB-[200-300]			RB-[800-1200]			RB-[2K-3K]		
	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓
Baseline Solvers									
KaMIS [57]	20.090*	0.000%	45.809s	43.004*	0.000%	56.974s	49.260*	0.000%	60.960s
Gurobi [57]	20.090	0.000%	0.538s	42.192	1.829%	33.843s	48.560	1.377%	3426.207s
GP-Degree	17.030	15.077%	0.000s	34.186	20.360%	0.004s	38.440	21.777%	0.019s
LC-Degree	18.480	7.916%	0.000s	37.778	12.058%	0.005s	42.540	13.569%	0.040s
Construction-based Solvers with Greedy Decoding									
GOAL [13]	18.268	8.959%	0.182s	37.596	12.476%	3.962s	42.640	13.304%	24.499s
GCN [15]	18.170	9.413%	0.014s	36.060	16.036%	0.048s	38.760	21.224%	0.240s
Meta-EGN [60]	17.692	11.778%	0.084s	34.670	19.259%	0.127s	38.840	21.039%	0.739s
VAG-CO [61]	17.308	13.709%	0.016s	34.002	20.779%	0.029s	37.260	24.195%	0.151s
DiffUCO [14]	19.200	4.369%	0.470s	38.490	10.428%	4.712s	42.120	14.352%	339.010s
Fast-T2T [13]	19.330	3.710%	0.192s	38.936	9.388%	0.714s	38.380	21.995%	8.980s
COExpander [5]	19.662	2.088%	0.112s	41.234	4.060%	0.624s	45.840	6.856%	5.340s
Search-based Solvers									
iSCO [42]#	19.29	3.98%	0.325s	36.96	14.05%	1.351s	–	–	–
FLD [48]#	20.02	0.35%	0.076s	40.25	6.40%	0.232s	–	–	–
RLD (SA) [32]	19.934	0.750%	0.076s	40.564	5.573%	0.140s	46.380	5.711%	0.375s
RLD (SA) (more) [32]	20.034	0.270%	0.112s	42.050	2.161%	0.499s	48.120	2.257%	3.309s
MCMC (fewer)	19.348	3.612±3.267%	0.085s	38.284	10.868±3.837%	0.155s	43.160	12.255±3.838%	0.850s
MCMC (more)	19.518	2.782±3.018%	0.170s	38.768	9.733±3.565%	0.453s	44.480	9.562±3.377%	5.261s
NLD-EG-GAT (fewer)	20.016	0.355±1.259%	0.092s	41.182	4.157±2.187%	0.165s	46.900	4.678±2.736%	0.415s
NLD-EG-GAT (more)	20.078	0.056±0.578%	0.117s	42.504	1.121±1.283%	0.502s	48.640	1.221±1.500%	3.378s
NLD-EG-GCN (fewer)	20.030	0.288±1.144%	0.093s	41.136	4.246±2.284%	0.168s	46.720	5.059±2.659%	0.416s
NLD-EG-GCN (more)	20.070	0.094±0.660%	0.119s	42.480	1.180±1.366%	0.514s	48.760	0.993±1.620%	3.331s

METHOD	ER-[700-800]			ER-[1400-1600]			ER-[9K-11K]		
	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓	OBJ.↑	DROP↓	TIME↓
Baseline Solvers									
KaMIS [57]	44.969*	0.000%	60.753s	50.938*	0.000%	60.824s	381.438*	0.000%	623.368s
Gurobi [57]	38.781	13.749%	60.489s	44.813	12.015%	3602.519s	–	–	–
GP-Degree	33.266	26.022%	0.002s	37.078	27.204%	0.007s	286.250	24.954%	0.523s
LC-Degree	38.852	13.595%	0.002s	44.109	13.395%	0.021s	342.063	10.323%	0.678s
Construction-based Solvers with Greedy Decoding									
GOAL [13]	40.477	9.986%	2.296s	45.594	10.485%	9.966s	–	–	–
GCN [15]	35.852	20.267%	0.039s	37.680	26.021%	0.117s	296.000	22.387%	1.229s
Meta-EGN [60]	33.148	26.280%	0.117s	37.555	26.262%	0.267s	–	–	–
VAG-CO [61]	33.813	24.804%	0.029s	36.914	27.533%	0.069s	287.375	24.658%	1.032s
Fast-T2T [13]	40.195	10.618%	0.641s	44.094	13.430%	5.117s	327.563	14.120%	27.553s
LwD [21]	37.289	17.068%	0.589s	39.523	22.400%	1.468s	–	–	–
COExpander [5]	42.383	5.746%	0.469s	48.164	5.432%	2.211s	366.750	3.851%	24.644s
Search-based Solvers									
iSCO [42]#	42.18	6.20%	0.680s	–	–	–	365.37	4.21%	247.50s
FLD [48]#	44.37	1.33%	0.148s	–	–	–	377.50	1.03%	4.20s
RLD (SA) [32]	44.211	1.682%	0.147s	49.797	2.227%	0.280s	375.625	1.522%	4.523s
RLD (SA) (more) [32]	44.977	-0.024%	0.249s	51.178	-0.475%	2.085s	377.688	0.982%	21.468s
MCMC (fewer)	39.844	11.389±2.556%	0.180s	44.844	11.951±2.391%	0.548s	346.938	9.041±1.081%	15.058s
MCMC (more)	40.359	10.245±2.614%	0.362s	46.016	9.654±2.412%	2.692s	352.125	7.683±0.888%	28.563s
NLD-EG-GAT (fewer)	44.359	1.349±1.416%	0.166s	50.023	1.781±1.705%	0.324s	–	–	–
NLD-EG-GAT (more)	45.078	-0.249±1.425%	0.258s	51.578	-1.270±1.498%	2.101s	–	–	–
NLD-EG-GCN (fewer)	44.453	1.139±1.684%	0.172s	50.031	1.768±1.598%	0.354s	378.875	0.668±0.543%	5.027s
NLD-EG-GCN (more)	45.078	-0.249±1.425%	0.264s	51.547	-1.208±1.527%	2.112s	383.938	-0.657±0.456%	23.067s

Table 15: Complete results on TSP.

METHOD	TSP-100			TSP-500			TSP-1K		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
Baseline Solvers									
Concorde [4]	7.756*	0.000%	0.238s	16.546*	0.000%	18.672s	23.118*	0.000%	84.413s
LKH (500) [59]	7.756	0.001%	0.176s	16.546	0.002%	1.848s	23.119	0.005%	4.641s
GA-EAX [58]	7.756	0.000%	1.862s	16.546	0.001%	1.857s	23.118	0.000%	17.544s
Nearest	9.680	24.798%	0.000s	20.808	25.766%	0.000s	28.903	25.023%	0.001s
Insertion [25]	8.509	9.722%	0.000s	18.552	12.125%	0.002s	26.116	12.972%	0.009s
Nearest + 2OPT	8.107	4.529%	0.000s	17.418	5.271%	0.012s	24.375	5.435%	0.085s
Insertion + 2OPT [25]	8.300	7.015%	0.000s	18.131	9.579%	0.009s	25.454	10.107%	0.054s
Construction-based Solvers with Greedy Decoding									
SymNCO [8]	7.872	1.491%	0.098s	–	–	–	–	–	–
GOAL [10]	7.905	1.920%	0.762s	18.006	8.825%	3.964s	25.380	9.789%	7.995s
GCN [15]	7.937	2.334%	0.007s	18.148	9.683%	0.047s	26.124	13.001%	0.195s
DIMES [16]	8.757	12.918%	0.005s	18.888	14.155%	0.273s	26.421	14.291%	0.546s
UTSP [19]	9.377	20.909%	0.009s	19.761	19.435%	0.059s	27.644	19.580%	0.223s
Fast-T2T [13]	7.774	0.230%	0.031s	17.473	5.603%	0.099s	25.021	8.229%	0.289s
COExpander [5]	7.761	0.073%	0.102s	17.119	3.466%	0.231s	24.698	6.834%	0.691s
Construction-based Solvers with Greedy Decoding and 2-OPT									
SymNCO + 2OPT [8]	7.840	1.087%	0.100s	–	–	–	–	–	–
GOAL + 2OPT [10]	7.862	1.369%	0.764s	17.047	3.027%	4.057s	23.806	2.974%	8.065s
GCN + 2OPT [15]	7.775	0.243%	0.009s	16.769	1.348%	0.063s	23.527	1.769%	0.227s
DIMES + 2OPT [16]	8.108	4.543%	0.012s	17.655	6.707%	0.314s	24.906	7.735%	0.662s
UTSP + 2OPT [19]	8.161	5.215%	0.017s	17.710	7.033%	0.090s	24.765	7.122%	0.290s
Fast-T2T + 2OPT [13]	7.762	0.078%	0.031s	16.690	0.869%	0.102s	23.423	1.319%	0.328s
Fast-T2T + 2OPT (×8) [13]	7.757	0.009±0.038%	0.118s	16.607	0.370±0.210%	0.591s	23.313	0.844±0.218%	2.471s
COExpander + 2OPT [5]	7.760	0.058%	0.102s	16.626	0.487%	0.242s	23.337	0.946%	0.703s
COExpander + 2OPT (×8) [5]	7.756	0.008±0.035%	0.329s	16.575	0.179±0.146%	1.265s	23.239	0.523±0.173%	5.104s
Construction-based Solvers with Greedy Decoding and MCTS									
GCN + MCTS [17]	7.762	0.073%	0.013s	16.658	0.676%	0.367s	23.380	1.131%	1.594s
DIMES + MCTS [16]	7.897	1.819%	0.025s	17.776	7.435%	0.833s	25.032	8.279%	3.870s
UTSP + MCTS [19]	8.052	3.817%	0.065s	17.913	8.261%	1.574s	25.069	8.440%	2.231s
D&C Solvers									
GLOP (fewer) [25]#	–	–	–	17.07	–	0.15s	24.01	–	0.27s
GLOP (more) [25]#	–	–	–	16.91	–	0.70s	23.84	–	1.41s
UDC [26]#	–	–	–	16.78	–	1.88s	23.53	–	3.75s
Search-based Solvers									
MCMC (fewer)	8.037	3.622±1.774%	0.035s	17.164	3.737±0.837%	0.178s	24.172	4.560±0.517%	0.359s
MCMC (fewer) (×8)	7.897	1.815±1.061%	0.097s	17.022	2.876±0.593%	0.297s	24.007	3.845±0.346%	0.555s
MCMC (more)	8.002	3.172±1.627%	0.071s	17.047	3.030±0.670%	0.355s	23.941	3.558±0.456%	0.725s
MCMC (more) (×8)	7.875	1.539±0.969%	0.143s	16.923	2.280±0.495%	0.539s	23.805	2.970±0.308%	1.039s
NLD-IG-GCN (fewer)	7.760	0.057±0.135%	0.075s	16.586	0.244±0.183%	0.320s	23.213	0.412±0.179%	0.679s
NLD-IG-GCN (fewer) (×8)	7.756	0.007±0.029%	0.136s	16.560	0.084±0.070%	0.829s	23.167	0.211±0.096%	1.673s
NLD-IG-GCN (more)	7.758	0.021±0.071%	0.129s	16.567	0.126±0.105%	0.800s	23.172	0.232±0.131%	1.685s
NLD-IG-GCN (more) (×8)	7.756	0.003±0.019%	0.235s	16.552	0.040±0.047%	2.029s	23.142	0.102±0.058%	4.066s
NLD-IG-GAT (fewer)	7.759	0.034±0.128%	0.034s	16.570	0.148±0.149%	0.129s	23.193	0.325±0.221%	0.189s
NLD-IG-GAT (fewer) (×8)	7.757	0.012±0.048%	0.078s	16.552	0.040±0.047%	0.234s	23.152	0.145±0.107%	0.549s
NLD-IG-GAT (more)	7.756	0.006±0.037%	0.092s	16.556	0.062±0.070%	0.304s	23.154	0.156±0.113%	0.428s
NLD-IG-GAT (more) (×8)	7.756	0.001±0.008%	0.139s	16.549	0.018±0.029%	0.546s	23.132	0.058±0.044%	1.278s

METHOD	TSP-2K			TSP-5K			TSP-10K		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
Baseline Solvers									
LKH (500) [59]	32.436*	0.000%	7.143s	50.968*	0.000%	34.494s	71.755*	0.000%	332.758s
Nearest	40.502	24.870%	0.003s	63.435	24.460%	0.015s	88.898	23.891%	0.061s
Insertion [25]	36.810	13.484%	0.036s	58.031	13.858%	0.225s	81.806	14.008%	0.846s
Nearest + Fast-2OPT	34.424	6.128%	0.845s	54.022	5.992%	1.396s	75.957	5.857%	2.554s
Insertion + Fast-2OPT [25]	35.817	10.423%	0.911s	56.412	10.683%	1.773s	79.563	10.882%	3.468s
Construction-based Solvers with Greedy Decoding									
GOAL [10]	–	–	–	–	–	–	82.249	14.627%	95.579s
GCN [15]	39.944	23.148%	0.879s	69.546	36.454%	1.671s	107.645	50.018%	26.409s
Fast-T2T [13]	38.118	17.519%	1.045s	65.607	28.730%	7.200s	102.972	43.504%	26.984s
COExpander [5]	37.003	14.082%	1.406s	63.217	24.037%	7.971s	97.423	35.772%	28.288s
Construction-based Solvers with Greedy Decoding and 2-OPT									
GOAL + 2OPT [10]	–	–	–	–	–	–	74.595	3.958%	102.369s
GCN + 2OPT [15]	33.028	1.829%	0.953s	51.929	1.889%	7.329s	73.019	1.761%	28.294s
Fast-T2T + 2OPT [13]	33.001	1.745%	1.101s	51.805	1.646%	7.698s	72.982	1.710%	28.313s
COExpander + 2OPT [5]	32.907	1.454%	1.448s	51.669	1.378%	8.393s	72.832	1.501%	33.485s
D&C Solvers and Search-based Solvers									
GLOP (fewer) [25]#	–	–	–	–	–	–	75.62	–	2.00s
GLOP (more) [25]#	–	–	–	–	–	–	75.29	–	6.75s
MCMC (fewer)	34.007	4.843±0.391%	1.822s	55.023	7.957±0.341%	7.499s	87.258	21.606±0.484%	22.874s
MCMC (more)	33.666	3.793±0.348%	3.640s	53.711	5.383±0.243%	19.201s	86.895	21.100±0.539%	68.844s
NLD-IG-GCN (fewer)	32.635	0.613±0.159%	0.829s	51.504	1.051±0.143%	4.709s	72.693	1.307±0.099%	19.056s
NLD-IG-GCN (more)	32.542	0.327±0.097%	2.085s	51.278	0.609±0.105%	13.983s	72.471	0.998±0.121%	55.948s

Table 16: Complete results on CVRP.

METHOD	CVRP-50			CVRP-100			CVRP-200		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
Baseline Solvers									
HGS [51]	10.366*	0.000%	1.005s	15.563*	0.000%	20.027s	19.630*	0.000%	60.024s
Nearest	13.981	35.245%	0.000s	20.902	34.718%	0.000s	26.467	35.201%	0.000s
Nearest + LS	11.397	9.913%	0.002s	16.947	8.877%	0.005s	21.082	7.490%	0.014s
Construction-based Solvers with Greedy Decoding									
SymNCO [8]	10.769	3.891%	0.087s	16.220	4.241%	0.166s	20.662	5.274%	0.320s
GOAL [10]	10.906	5.193%	0.504s	16.342	5.005%	0.962s	21.439	9.263%	1.978s
Fast-T2T [13]	12.640	21.835%	0.009s	19.202	23.333%	0.010s	25.064	27.616%	0.059s
COExpander [5]	11.979	15.407%	0.033s	17.497	12.343%	0.047s	22.402	13.977%	0.145s
Construction-based Solvers with Greedy Decoding and Classic-LS									
SymNCO + LS [8]	10.565	1.910%	0.092s	15.933	2.379%	0.173s	20.193	2.880%	0.341s
GOAL + LS [10]	10.628	2.519%	0.507s	15.959	2.548%	0.969s	20.396	3.936%	2.037s
Fast-T2T + LS [13]	10.871	4.836%	0.013s	16.294	4.698%	0.018s	20.662	5.290%	0.063s
Fast-T2T + LS (×8) [13]	10.697	3.180%	0.038s	16.112	3.527%	0.103s	20.404	3.956%	0.313s
COExpander + LS [5]	10.773	3.903%	0.037s	16.224	4.253%	0.055s	20.587	4.893%	0.153s
COExpander + LS (×8) [5]	10.572	1.983%	0.124s	15.945	2.455%	0.312s	20.267	3.256%	0.937s
Search-based Solvers									
NeuOpt-GIRE (T=1k) [29]#	10.430	0.61%	0.072s	15.865	1.94%	0.168s	–	–	–
NeuOpt-GIRE (T=5k) [29]#	10.382	0.16%	0.354s	15.698	0.87%	0.828s	–	–	–
MCMC (more)	–	–	–		–	–	20.647	5.171±1.891%	0.716s
MCMC (more) (×8)	–	–	–		–	–	20.310	3.446±0.902%	0.962s
NLD-IG-GCN (fewer)	10.501	1.301±1.286%	0.019s	15.795	1.504±0.969%	0.038s	19.962	1.698±0.690%	0.140s
NLD-IG-GCN (fewer) (×8)	10.407	0.389±0.575%	0.054s	15.663	0.649±0.491%	0.126s	19.832	1.033±0.466%	0.368s
NLD-IG-GCN (more)	10.393	0.253±0.450%	0.079s	15.638	0.482±0.452%	0.169s	19.795	0.840±0.505%	0.540s
NLD-IG-GCN (more) (×8)	10.372	0.057±0.189%	0.211s	15.589	0.166±0.231%	0.428s	19.709	0.405±0.306%	1.230s
NLD-IG-GAT (fewer)	10.470	0.991±1.119%	0.029s	15.748	1.191±0.829%	0.043s	19.909	1.431±0.732%	0.099s
NLD-IG-GAT (fewer) (×8)	10.401	0.337±0.510%	0.047s	15.646	0.534±0.431%	0.075s	19.773	0.731±0.420%	0.228s
NLD-IG-GAT (more)	10.391	0.236±0.416%	0.104s	15.628	0.413±0.395%	0.180s	19.747	0.604±0.463%	0.419s
NLD-IG-GAT (more) (×8)	10.373	0.066±0.203%	0.181s	15.589	0.164±0.229%	0.290s	19.685	0.281±0.238%	0.873s

METHOD	CVRP-500			CVRP-1K			CVRP-2K		
	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓	OBJ.↓	DROP↓	TIME↓
Baseline Solvers									
HGS [51]	37.154*	0.000%	360.376s	41.171*	0.000%	360.051s	57.181*	0.000%	360.302s
Nearest	47.427	27.981%	0.000s	53.275	26.171%	0.001s	72.375	23.545%	0.003s
Nearest + LS	40.127	8.118%	0.075s	45.103	6.614%	0.285s	61.858	5.471%	1.309s
Construction-based Solvers with Greedy Decoding									
SymNCO [8]	40.382	8.723%	0.769s	–	–	–	–	–	–
GOAL [10]	38.846	4.587%	7.095s	–	–	–	–	–	–
Fast-T2T [13]	47.749	28.509%	0.091s	–	–	–	–	–	–
COExpander [5]	43.901	18.199%	0.554s	–	–	–	–	–	–
Construction-based Solvers with Greedy Decoding and Classic-LS									
SymNCO + LS [8]	38.700	4.173%	0.883s	–	–	–	–	–	–
GOAL + LS [10]	38.118	2.620%	7.312s	–	–	–	–	–	–
Fast-T2T + LS [13]	39.195	5.530%	0.215s	–	–	–	–	–	–
Fast-T2T + LS (×8) [13]	38.792	4.441%	1.610s	–	–	–	–	–	–
COExpander + LS [5]	39.121	5.337%	0.605s	–	–	–	–	–	–
COExpander + LS (×8) [5]	38.649	4.044%	4.757s	–	–	–	–	–	–
D&C Solvers									
TAM-LKH [62]#	–	–	–	46.30	12.46%	1.80s	64.80	13.32%	5.60s
GLOP-LKH [25]#	–	–	–	45.90	11.49%	1.10s	63.00	10.18%	1.50s
UDC [26]#	–	–	–	43.48	5.61%	8.40s	60.94	6.57%	13.80s
Search-based Solvers									
MCMC (more)	39.322	5.627±1.076%	3.405s	43.877	6.649±1.378%	6.762s	60.203	5.426±1.692%	31.523s
MCMC (more) (×8)	38.842	4.543±0.528%	4.192s	–	–	–	–	–	–
NLD-IG-GCN (fewer)	38.047	2.404±0.519%	0.654s	–	–	–	–	–	–
NLD-IG-GCN (fewer) (×8)	37.839	1.849±0.336%	1.862s	–	–	–	–	–	–
NLD-IG-GCN (more)	37.697	1.463±0.399%	2.500s	–	–	–	–	–	–
NLD-IG-GCN (more) (×8)	37.530	1.011±0.260%	6.739s	–	–	–	–	–	–
NLD-IG-GAT (fewer)	37.959	2.177±0.526%	0.319s	42.852	4.119±0.761%	1.322s	59.428	4.020±1.189%	5.372s
NLD-IG-GAT (fewer) (×8)	37.781	1.681±0.388%	0.938s	–	–	–	–	–	–
NLD-IG-GAT (more)	37.660	1.364±0.385%	1.432s	42.304	2.781±0.637%	6.212s	58.327	2.093±1.064%	24.835s
NLD-IG-GAT (more) (×8)	37.494	0.916±0.283%	3.846s	–	–	–	–	–	–

Table 17: Generalization Study on TSPLIB.

INSTANCE	GCN100	GCN500	GCN1K	GAT100	GAT500	GAT1K	BEST
eil51	0.000%	0.026%	0.000%	0.000%	0.000%	0.000%	0.000%
berlin52	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
st70	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
eil76	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
pr76	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
rat99	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
kroA100	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
kroB100	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
kroC100	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
kroD100	0.000%	0.000%	0.000%	0.000%	0.000%	0.072%	0.000%
kroE100	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
rd100	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
eil101	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
lin105	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
pr107	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
pr124	0.000%	0.301%	0.075%	0.000%	0.000%	0.000%	0.000%
bier127	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
ch130	0.000%	0.000%	0.000%	0.000%	0.000%	0.235%	0.000%
pr136	0.000%	0.000%	0.015%	0.010%	0.015%	0.025%	0.000%
pr144	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
ch150	0.000%	0.000%	0.000%	0.000%	0.045%	0.000%	0.000%
kroA150	0.001%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
kroB150	0.001%	0.000%	0.000%	0.050%	0.050%	0.000%	0.000%
pr152	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
u159	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
rat195	0.183%	0.142%	0.183%	0.114%	0.114%	0.118%	0.114%
d198	0.000%	0.006%	0.017%	0.006%	0.016%	0.005%	0.000%
kroA200	0.000%	0.000%	0.000%	0.000%	0.000%	0.047%	0.000%
kroB200	0.000%	0.000%	0.000%	0.003%	0.000%	0.334%	0.000%
ts225	0.131%	0.131%	0.262%	0.131%	0.196%	0.000%	0.000%
tsp225	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
pr226	0.000%	0.010%	0.010%	0.000%	0.000%	0.000%	0.000%
gil262	0.000%	0.018%	0.035%	0.018%	0.039%	0.039%	0.000%
pr264	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
a280	2.506%	0.000%	0.128%	2.414%	0.040%	0.995%	0.000%
pr299	0.011%	0.191%	0.299%	0.000%	0.000%	0.000%	0.000%
lin318	0.000%	0.000%	0.000%	0.260%	0.000%	0.182%	0.000%
rd400	0.070%	0.032%	0.116%	0.206%	0.000%	0.090%	0.000%
fl417	0.130%	0.000%	0.060%	0.093%	0.118%	0.020%	0.000%
pr439	0.081%	0.031%	0.031%	0.118%	0.085%	0.207%	0.031%
pcb442	0.317%	0.238%	0.273%	0.370%	0.013%	0.119%	0.013%
d493	0.294%	0.287%	0.259%	0.275%	0.486%	0.596%	0.259%
u574	0.563%	0.000%	0.019%	0.506%	0.000%	0.106%	0.000%
rat575	0.690%	0.032%	0.051%	0.398%	0.013%	0.021%	0.013%
p654	0.095%	0.032%	0.025%	0.025%	0.079%	0.043%	0.025%
d657	0.521%	0.044%	0.005%	0.768%	0.083%	0.013%	0.005%
u724	0.511%	0.108%	0.129%	0.599%	0.054%	0.066%	0.054%
rat783	0.452%	0.023%	0.023%	0.958%	0.114%	0.027%	0.023%
pr1002	1.280%	0.234%	0.244%	1.230%	0.374%	0.012%	0.012%
mean obj.	8.081	8.066	8.067	8.083	8.066	8.068	**8.063**
mean drop	0.160%	0.039%	0.046%	0.175%	0.039%	0.068%	**0.011**%

Table 18: Generalization Study on CVRPLIB.

INSTANCE	GCN-100	GCN-200	GCN-500	GAT-100	GAT-200	GAT-500	BEST
X-n101-k25	0.344%	1.033%	1.252%	1.139%	0.022%	1.539%	0.022%
X-n106-k14	0.293%	0.245%	0.228%	0.228%	0.125%	0.245%	0.125%
X-n110-k13	-0.008%	-0.008%	-0.008%	-0.008%	-0.008%	-0.008%	-0.008%
X-n115-k10	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
X-n120-k6	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
X-n125-k30	1.847%	2.507%	1.903%	1.688%	1.836%	1.600%	1.600%
X-n129-k18	0.282%	0.120%	0.196%	0.210%	0.028%	0.054%	0.028%
X-n134-k13	0.169%	0.131%	0.300%	0.300%	0.012%	0.231%	0.012%
X-n139-k10	0.091%	0.110%	0.140%	0.005%	-0.002%	0.110%	-0.002%
X-n143-k7	0.194%	0.276%	0.168%	0.170%	0.210%	0.168%	0.168%
X-n148-k46	0.991%	0.687%	0.915%	0.957%	1.014%	0.812%	0.687%
X-n153-k22	1.089%	0.902%	0.994%	0.770%	0.750%	1.115%	0.750%
X-n157-k13	-0.007%	-0.007%	-0.007%	-0.007%	-0.007%	-0.007%	-0.007%
X-n162-k11	0.372%	0.293%	0.293%	0.005%	0.261%	0.290%	0.005%
X-n167-k10	-0.005%	0.152%	0.118%	0.378%	0.174%	0.118%	-0.005%
X-n172-k51	1.602%	1.604%	1.142%	1.388%	0.507%	1.785%	0.507%
X-n176-k26	1.547%	1.283%	0.946%	2.071%	0.893%	1.900%	0.893%
X-n181-k23	0.108%	0.200%	0.193%	0.076%	0.141%	0.118%	0.076%
X-n186-k15	0.034%	0.065%	0.108%	0.034%	0.113%	0.219%	0.034%
X-n190-k8	0.369%	0.471%	0.360%	0.572%	0.699%	0.333%	0.333%
X-n195-k51	1.496%	1.370%	1.318%	1.285%	1.000%	1.508%	1.000%
X-n200-k36	2.265%	1.939%	2.138%	2.165%	2.032%	1.945%	1.939%
X-n204-k19	0.928%	1.000%	0.774%	0.610%	0.759%	0.108%	0.108%
X-n209-k16	0.264%	0.128%	0.413%	0.216%	0.266%	0.225%	0.128%
X-n214-k11	1.193%	0.850%	1.276%	1.181%	0.727%	0.946%	0.727%
X-n219-k73	0.155%	0.082%	0.125%	0.184%	0.235%	0.152%	0.082%
X-n223-k34	1.454%	0.975%	1.195%	1.045%	1.223%	1.024%	0.975%
X-n228-k23	1.118%	1.098%	0.758%	0.499%	0.319%	0.797%	0.319%
X-n233-k16	0.444%	0.744%	0.710%	1.004%	0.718%	0.971%	0.444%
X-n237-k14	0.022%	0.030%	0.222%	0.046%	-0.007%	0.197%	-0.007%
X-n242-k48	1.044%	1.426%	0.991%	1.428%	0.906%	1.684%	0.906%
X-n247-k50	2.123%	3.002%	3.844%	1.651%	2.494%	1.193%	1.193%
X-n251-k28	0.708%	0.867%	0.654%	0.833%	0.848%	0.745%	0.654%
X-n256-k16	0.211%	0.400%	0.294%	0.408%	0.242%	0.211%	0.211%
X-n261-k13	1.180%	0.610%	0.456%	1.300%	0.782%	1.198%	0.456%
X-n266-k58	1.356%	1.287%	1.763%	1.901%	1.609%	1.483%	1.287%
X-n270-k35	0.750%	0.807%	0.631%	0.882%	0.805%	0.573%	0.573%
X-n275-k28	0.238%	0.658%	0.576%	0.611%	0.416%	0.730%	0.238%
X-n280-k17	0.933%	0.606%	0.906%	0.661%	0.958%	0.887%	0.606%
X-n284-k15	1.547%	1.477%	1.499%	1.655%	1.509%	1.410%	1.410%
X-n289-k60	1.805%	2.178%	2.218%	1.393%	1.880%	1.546%	1.393%
X-n294-k50	1.741%	0.856%	1.751%	1.459%	1.822%	1.687%	0.856%
X-n298-k31	1.287%	1.035%	0.395%	1.090%	1.077%	1.551%	0.395%
X-n303-k21	0.709%	0.781%	0.958%	1.068%	0.984%	1.055%	0.709%
X-n308-k13	0.954%	0.981%	0.619%	0.510%	0.700%	0.787%	0.510%
X-n313-k71	2.257%	2.118%	1.840%	2.190%	1.898%	1.944%	1.840%
X-n317-k53	0.158%	0.234%	0.214%	0.306%	0.105%	0.258%	0.105%
X-n322-k28	1.192%	1.192%	1.315%	0.917%	1.518%	1.544%	0.917%
X-n327-k20	0.747%	0.949%	0.875%	1.210%	0.772%	0.774%	0.747%
X-n331-k15	0.209%	0.328%	0.010%	0.161%	0.197%	0.206%	0.010%
X-n336-k84	2.500%	2.498%	2.669%	2.301%	1.988%	2.049%	1.988%
X-n344-k43	1.278%	1.799%	1.759%	1.853%	1.308%	1.626%	1.278%
X-n351-k40	1.527%	1.905%	1.928%	1.955%	1.828%	1.944%	1.527%
X-n359-k29	1.049%	1.131%	0.981%	1.171%	1.234%	1.353%	0.981%
X-n367-k17	1.013%	1.167%	1.305%	1.363%	1.484%	1.388%	1.013%
X-n376-k94	0.381%	0.359%	0.248%	0.470%	0.412%	0.321%	0.248%
X-n384-k52	1.621%	1.729%	1.798%	1.653%	1.549%	1.538%	1.538%
X-n393-k38	1.302%	1.462%	1.354%	1.537%	1.218%	1.185%	1.185%
X-n401-k29	0.760%	0.787%	0.671%	0.796%	1.003%	0.876%	0.671%
X-n411-k19	1.034%	0.763%	1.004%	1.071%	1.183%	0.963%	0.763%
X-n420-k130	2.505%	1.913%	2.365%	2.117%	2.129%	1.965%	1.913%
X-n429-k61	1.697%	1.865%	1.998%	1.897%	2.088%	2.073%	1.697%
X-n439-k37	1.021%	1.157%	0.973%	1.171%	1.198%	1.122%	0.973%
X-n449-k29	1.473%	1.681%	1.708%	1.730%	1.868%	2.072%	1.473%
X-n459-k26	1.001%	0.607%	0.656%	1.195%	1.154%	1.328%	0.607%
X-n469-k138	3.318%	3.447%	3.658%	2.608%	2.902%	3.098%	2.608%
X-n480-k70	1.430%	1.796%	2.023%	2.011%	1.700%	2.115%	1.430%
X-n491-k59	1.930%	1.895%	1.619%	1.692%	1.772%	2.538%	1.619%
X-n502-k39	0.307%	0.270%	0.256%	0.381%	0.350%	0.285%	0.256%
X-n513-k21	1.049%	1.289%	0.956%	0.838%	1.307%	1.133%	0.838%
mean obj.	45.776	45.787	45.797	45.765	45.741	45.780	**45.645**
mean drop	0.971%	0.994%	0.999%	0.995%	0.932%	1.013%	**0.722%**

Table 19: NLD4CO vs. Previous SOTA Solvers.

Dataset	Size	Baseline Solvers			Previous SOTA Solvers			NLD4CO	
		Method	Obj.	Time↓	Method	Drop↓	Time↓	Drop↓	Time↓
MIS: Compared with NLD-EG (more)									
RB-[200-300]	500	KaMIS [57]	20.090	45.809s	COExpander [5]	2.088%	0.112s	0.056%	0.117s
RB-[800-1200]	500	KaMIS [57]	43.004	56.974s	COExpander [5]	4.060%	0.624s	1.180%	0.514s
RB-[2K-3K]	50	KaMIS [57]	49.260	60.960s	COExpander [5]	6.856%	5.340s	0.993%	3.331s
ER-[700-800]	128	KaMIS [57]	44.969	60.753s	COExpander [5]	5.746%	0.469s	-0.249%	0.264s
ER-[1400-1600]	128	KaMIS [57]	50.938	60.824s	COExpander [5]	5.432%	2.211s	-1.270%	2.101s
ER-[9K-11K]	16	KaMIS [57]	381.438	623.368s	COExpander [5]	3.851%	24.644s	-0.657%	23.067s
MWIS: Compared with NLD-EG (more)									
RB-[200-300]	500	Gurobi [3]	14.856	0.192s	Fast-T2T [13]	2.851%	0.166s	0.043%	0.118s
RB-[800-1200]	500	Gurobi [3]	32.927	39.819s	Fast-T2T [13]	14.909%	0.371s	0.272%	0.496s
RB-[2K-3K]	50	Gurobi [3]	37.971	53.798s	Fast-T2T [13]	19.537%	3.827s	-1.623%	3.117s
TSP: Compared with NLD-IG (fewer)									
TSP-100	1280	Concorde [4]	7.756	0.238s	COExpander [5]	0.008%	0.329s	0.007%	0.136s
TSP-500	128	Concorde [4]	16.546	18.672s	COExpander [5]	0.179%	1.265s	0.040%	0.234s
TSP-1K	128	Concorde [4]	23.118	84.413s	COExpander [5]	0.523%	5.104s	0.145%	0.549s
TSP-2K	64	LKH [59]	32.436	7.143s	COExpander [5]	1.454%	1.448s	0.613%	0.829s
TSP-5K	32	LKH [59]	50.968	34.494s	COExpander [5]	1.378%	8.393s	1.051%	4.709s
TSP-10K	16	LKH [59]	71.755	332.758s	COExpander [5]	1.501%	33.485s	1.307%	19.056s
CVRP: Compared with NLD-IG (fewer)									
CVRP-50	10000	HGS [51]	10.366	1.005s	SymNCO [8]	1.910%	0.092s	0.337%	0.047s
CVRP-100	10000	HGS [51]	15.563	20.027s	SymNCO [8]	2.379%	0.173s	0.534%	0.075s
CVRP-200	100	HGS [51]	19.630	60.024s	SymNCO [8]	2.880%	0.341s	0.731%	0.228s
CVRP-500	100	HGS [51]	37.154	360.376s	GOAL [10]	2.620%	7.312s	1.681%	0.938s
CVRP-1K	100	HGS [51]	41.171	360.051s	UDC [26]	5.608%	8.400s	4.119%	1.322s
CVRP-2K	100	HGS [51]	57.181	360.302s	UDC [26]	6.574%	13.800s	4.020%	5.372s
Summary: NLD4CO compared with previous construction-based and D&C SOTA solvers									
MIS	Summary	Baseline	–	**151.448s**	Previous SOTA	**4.672%**	**5.567s**	**0.009%**	**4.899s**
MWIS	Summary	Baseline	–	**31.270s**	Previous SOTA	**12.432%**	**1.455s**	**-0.436%**	**1.244s**
TSP	Summary	Baseline	–	**79.620s**	Previous SOTA	**0.841%**	**8.337s**	**0.527%**	**4.252s**
CVRP	Summary	Baseline	–	**193.631s**	Previous SOTA	**3.662%**	**5.020s**	**1.904%**	**1.330s**

Dataset	Size	Baseline Solvers			Previous SOTA Solvers			NLD4CO	
		Method	Obj.	Time↓	Method	Drop↓	Time↓	Drop↓	Time↓
MIS: Compared with NLD-EG (fewer)									
RB-[200-300]	500	KaMIS [57]	20.090	45.809s	FLD [48]	0.350%	0.076s	0.288%	0.093s
RB-[800-1200]	500	KaMIS [57]	43.004	56.974s	RLD [32]	5.573%	0.140s	4.157%	0.165s
RB-[2K-3K]	50	KaMIS [57]	49.260	60.960s	RLD [32]	5.711%	0.375s	4.678%	0.415s
ER-[700-800]	128	KaMIS [57]	44.969	60.753s	FLD [48]	1.330%	0.148s	1.139%	0.172s
ER-[1400-1600]	128	KaMIS [57]	50.938	60.824s	RLD [32]	2.227%	0.280s	1.768%	0.354s
ER-[9K-11K]	16	KaMIS [57]	381.438	623.368s	FLD [48]	1.030%	4.200s	0.668%	5.027s
MWIS: Compared with NLD-EG (fewer)									
RB-[200-300]	500	Gurobi [3]	14.856	0.192s	RLD [32]	1.053%	0.078s	0.262%	0.094s
RB-[800-1200]	500	Gurobi [3]	32.927	39.819s	RLD [32]	7.440%	0.137s	3.970%	0.169s
RB-[2K-3K]	50	Gurobi [3]	37.971	53.798s	RLD [32]	7.274%	0.295s	3.617%	0.398s
TSP: Compared with NLD-IG (more)									
TSP-100	1280	Concorde [4]	7.756	0.238s	MCMC [43]	1.539%	0.143s	0.001%	0.139s
TSP-500	128	Concorde [4]	16.546	18.672s	MCMC [43]	2.280%	0.539s	0.018%	0.546s
TSP-1K	128	Concorde [4]	23.118	84.413s	MCMC [43]	2.970%	1.039s	0.058%	1.278s
TSP-2K	64	LKH [59]	32.436	7.143s	MCMC [43]	3.793%	3.640s	0.327%	2.085s
TSP-5K	32	LKH [59]	50.968	34.494s	MCMC [43]	5.383%	19.201s	0.609%	13.983s
TSP-10K	16	LKH [59]	71.755	332.758s	MCMC [43]	21.100%	68.844s	0.998%	55.948s
CVRP: Compared with NLD-IG (more)									
CVRP-50	10000	HGS [51]	10.366	1.005s	NeuOpt-GIRE [29]#	0.160%	0.354s	0.057%	0.211s
CVRP-100	10000	HGS [51]	15.563	20.027s	NeuOpt-GIRE [29]#	0.870%	0.828s	0.164%	0.290s
CVRP-200	100	HGS [51]	19.630	60.024s	MCMC [43]	3.446%	0.962s	0.281%	0.873s
CVRP-500	100	HGS [51]	37.154	360.376s	MCMC [43]	4.543%	4.192s	0.916%	3.846s
CVRP-1K	100	HGS [51]	41.171	360.051s	MCMC [43]	6.649%	6.762s	2.781%	6.212s
CVRP-2K	100	HGS [51]	57.181	360.302s	MCMC [43]	5.426%	31.523s	2.093%	24.835s
Summary: NLD4CO compared with previous search-based SOTA solvers									
MIS	Summary	Baseline	–	**151.448s**	Previous SOTA	**2.704%**	**0.870s**	**2.116%**	**1.038s**
MWIS	Summary	Baseline	–	**31.270s**	Previous SOTA	**5.256%**	**0.170s**	**2.616%**	**0.220s**
TSP	Summary	Baseline	–	**79.620s**	Previous SOTA	**6.178%**	**15.568s**	**0.335%**	**12.330s**
CVRP	Summary	Baseline	–	**193.631s**	Previous SOTA	**3.516%**	**7.437s**	**1.049%**	**6.045s**

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 abstract and introduction explicitly state the claims made, including the contributions made in the paper (Sec. 1). The claims match the experimental results in Sec. 5.

   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: We discuss the limitations in Appendix E.

   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: This 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: The experimental details, model settings, and solving settings are in Sec. 5 and Appendix F. The code is available at https://github.com/Thinklab-SJTU/NLD4CO.

   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: Code is available at https://github.com/Thinklab-SJTU/NLD4CO.

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: The experimental details are in Sec. 5 and Appendix F.

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 follow ML4CO-Bench-101 [7] to include the standard deviation of the data in the complete experimental tables in Appendix C and Appendix F.

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: We provide the computational environment required for the experiments in Appendix F. Further, for each experimental result, we also provide the time needed under the given environment.

   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: The research conducted in the paper conform, in every respect, 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: We discuss the broader impacts in Appendix E.

    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: The original papers that introduce models and datasets used in the paper are cited in Sec. 5.

    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/NLD4CO, 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: This paper does not incur such risks.

    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.

    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.