
StruDiCO: Structured Denoising Diffusion with Gradient-free Inference-stage Boosting for Memory and Time Efficient Combinatorial Optimization

Yu Wang¹, Yang Li², Junchi Yan^{2*}, Yi Chang^{1,3,4*}

¹School of Artificial Intelligence, Jilin University

²School of Artificial Intelligence, Shanghai Jiao Tong University

³International Center of Future Science, Jilin University

⁴Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, MOE, China
{yu_w, yichang}@jlu.edu.cn {yanglily, yanjunchi}@sjtu.edu.cn

Abstract

Diffusion models have recently emerged as powerful neural solvers for combinatorial optimization (CO). However, existing approaches fail to reveal how variables are progressively determined during inference, making the final solution opaque until the last step. To address this limitation, we propose a structured denoising diffusion model, StruDiCO, which incrementally constructs solutions through step-wise variable selection. This is achieved via a variable-absorption noising model, wherein the forward process simulates gradual variable deactivation, converging to an empty solution, while the reverse process incrementally selects variables to reconstruct the final solution. This design induces structural continuity across intermediate states, enabling interpretable and trajectory-consistent partial solutions throughout inference. To further improve the reliability of reverse inference, we introduce a constrained consistency sampling strategy, which suppresses low-confidence variable selection at each step to stabilize the reverse process. Leveraging the structure-preserving reverse process, we further propose a lightweight, gradient-free, objective-aware refinement framework, which iteratively improves solution quality by applying structure-aware perturbations to the current solution, performing reverse inference through the constraint consistency model, and decoding with an objective-guided scoring scheme. Extensive experiments on two canonical CO tasks, the Traveling Salesman Problem (TSP) and Maximal Independent Set (MIS), show that StruDiCO outperforms state-of-the-art diffusion-based solvers, achieving up to $3.5\times$ faster inference, 70% lower GPU memory usage, and significantly improved solution quality, with up to 37.7% drop reduction on TSP and an average 38.1% improvement on MIS. The codes are publicly available at <https://github.com/yuuuwang/StruDiCO>.

1 Introduction

Combinatorial optimization (CO) entails optimizing discrete decision variables toward specified objectives and serves as a cornerstone for numerous practical applications requiring efficient decision-making [1, 2, 3]. However, the inherent NP-hardness of CO problems renders large-scale CO instances particularly challenging to solve efficiently, traditionally necessitating the design of handcrafted

* Corresponding authors: Junchi Yan and Yi Chang (yanjunchi@sjtu.edu.cn; yichang@jlu.edu.cn). This work was partly supported by MOST of China (Grant 2023YFF0905400), NSFC (Grant U2341229, 92370201), and the China Postdoctoral Science Foundation Fellowship (Grant GZC20230947).

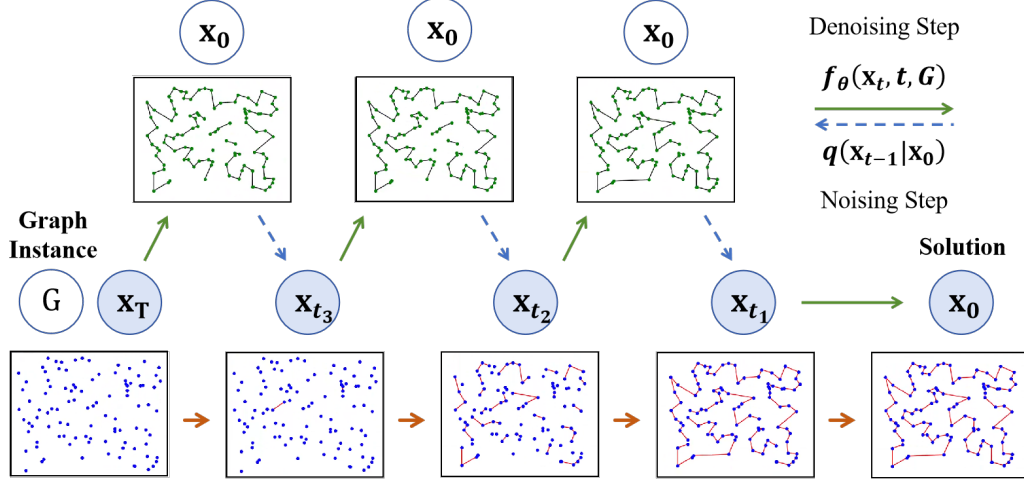


Figure 1: Overview of the structured denoising diffusion framework for combinatorial optimization. *Structured* denotes the preservation and progressive evolution of solution structures across timesteps. The forward process ($0 \rightarrow T$) gradually removes selected edges through a variable-absorption noising model, while the reverse process incrementally reconstructs the solution by selecting edges.

heuristics. Recently, neural combinatorial optimization (NCO), including both direct solvers [4, 5, 6, 7, 8] and methods that support solving [9, 10, 11], has emerged as a data-driven paradigm, offering notable improvements in both solution quality and computational efficiency.

Existing NCO solvers can be broadly categorized based on their solution construction paradigm: autoregressive and non-autoregressive. **Autoregressive constructive solvers** [12, 13, 14, 15, 16], formulates the task as an n -step sequential variable selection problem, where at each step the model produces a prediction heatmap over variables, conditioning on the current partial solution to decide the next inclusion. While this step-wise generation enables fine-grained control and interpretability, it incurs significant computational overhead due to its inherently sequential nature, limiting scalability to larger instances. In contrast, **non-autoregressive solvers** [17, 18, 19, 20, 21] adopt a one-shot generation strategy. These models directly generate neural heatmaps indicating the predicted likelihood of each variable being part of the solution, followed by heuristic decoding to enforce feasibility. Among them, generative models, such as diffusion-based models [22, 23, 24] have shown strong performance by modeling instance-conditioned distribution over high-quality solutions through a sequential denoising process. However, these models still suffer from *opaque intermediate states*, providing little insight into how the final solution is gradually formed—since decisions are not explicitly revealed until the final step. **In summary**, autoregressive models provide step-wise interpretability but suffer from sequential computational overhead, whereas non-autoregressive generative models offer efficient global modeling but lack transparency in the solution trajectory. Bridging this gap is essential for developing NCO solvers that are both interpretable and efficient.

To bridge the gap between step-wise interpretability and global distribution modeling, we propose a structured discrete diffusion framework tailored for CO. Specifically, we introduce a *variable-absorption noising model*, which progressively deactivates variables during the forward process, ensuring that each intermediate state corresponds to a structurally valid partial solution. Rather than treating such interpretability as an end in itself, this structure-preserving design enables more informed and reliable decision-making throughout the inference trajectory. To enhance the reliability of reverse inference, we introduce a *constrained consistency sampling* strategy that masks low-confidence variables at each step. By narrowing the effective search space, this strategy improves structural continuity across intermediate states and stabilizes the solution trajectory. Building on this structure-preserving foundation, we propose a *lightweight gradient-free, objective-aware refinement framework* that directly incorporates task objectives into the decoding process. Unlike prior refinement methods [23, 24] that rely on gradient-based search over relaxed continuous objectives, our approach eliminates the need for backward computation entirely. The refinement proceeds in three stages: it first perturbs a feasible solution using structure-aware noise, then performs reverse inference via the consistency model, and finally reconstructs a new solution through objective-guided greedy decoding. Together, these innovations yield a diffusion-based solver that achieves high-quality solutions with

significantly reduced computational overhead, while preserving structural interpretability and aligning inference with task objectives, effectively combining the complementary strengths of autoregressive and non-autoregressive paradigms. **Our contributions toward building an interpretable and efficient neural solver for CO are summarized as follows:**

- We propose a structure-preserving discrete diffusion framework and develop StruDiCO, which leverages variable-absorption forward diffusion and constrained reverse sampling to produce interpretable, high-quality solutions for combinatorial optimization tasks.
- We introduce a gradient-free, objective-aware refinement framework that efficiently incorporates task objectives without requiring backward gradient computation [23, 24].
- We evaluate our method on TSP and MIS tasks, demonstrating superior solution quality, $3.5\times$ faster inference, and 70% lower memory usage compared to state-of-the-art neural solvers.

2 Preliminaries, Problem Definition and Related Works

2.1 Combinatorial Optimization on Graphs

Following the notations adopted in [22, 23, 24], we define $\mathcal{G}$ as the universe of CO problem instances represented by graphs $G(V, E) \in \mathcal{G}$, where V and E denote the node set and edge set respectively. CO problems can be broadly classified into two types based on the solution composition: edge-selecting problems which involve selecting a subset of edges, and node-selecting problems which select a subset of nodes, both subject to the feasibility constraints of the problem. Let $\mathbf{x} \in \{0, 1\}^N$ denote the optimization variable. For edge-selecting problems, $N = n^2$ and $\mathbf{x}_{i \cdot n + j}$ indicates whether E_{ij} is included in $\mathbf{x}$. For node-selecting problems, $N = n$ and $\mathbf{x}_i$ indicates whether V_i is included in $\mathbf{x}$. The feasible set Ω consists of $\mathbf{x}$ satisfying specific constraints as feasible solutions. A CO problem on G aims to find a feasible $\mathbf{x}$ that minimize the given objective function $l(\cdot; G) : \{0, 1\}^N \rightarrow \mathbb{R}_{\geq 0}$:

$$\min_{\mathbf{x} \in \{0, 1\}^N} l(\mathbf{x}; G) \text{ s.t. } \mathbf{x} \in \Omega$$

In this paper, we study two primary and representative CO problems: TSP and MIS. TSP defines on an undirected complete graph $G = (V, E)$, where V represents n cities and each edge E_{ij} has a non-negative weight w_{ij} representing the distance between cities i and j . The problem can then be formulated as finding a Hamiltonian cycle of minimum weight in G . For MIS, given an undirected graph $G = (V, E)$, an independent set is a subset of vertices $S \subset V$ such that no two vertices in S are adjacent in G . MIS is the problem of finding an independent set of maximum cardinality in G .

2.2 Optimization Consistency

Given a problem instance G and its corresponding optimal solution $\mathbf{x}^*$, the discrete diffusion noising process defines a trajectory $\mathbf{x}_{0:T} = \mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T$, where $\mathbf{x}_0 = \mathbf{x}^*$ and each $\mathbf{x}_t$ is sampled from the conditional distribution $q(\mathbf{x}_t | \mathbf{x}_0)$. In standard diffusion models, generating samples requires modeling the reverse transitions $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, G)$ through multiple iterative steps, which can be computationally expensive. To enable efficient inference, *Consistency Models* [25] introduce the notion of self-consistency, where the model learns to map any noised state $\mathbf{x}_t$ at time t directly back to the clean source $\mathbf{x}_0$. Building on this idea, Fast T2T [24] gives the notion of *optimization consistency* for CO, in which all states along a noising trajectory are mapped to the same optimal solution $\mathbf{x}^*$ conditioned on the instance G . Formally, the consistency function $f_\theta(\mathbf{x}_t, t, G)$, parameterized by θ , maps a noisy input to a denoised prediction. Under self-consistency, the training objective is to minimize the discrepancy between any two mappings along the same trajectory:

$$\mathcal{L}(\theta) = \mathbb{E}_{t_1, t_2} [d(f_\theta(\mathbf{x}_{t_1}, t_1, G), f_\theta(\mathbf{x}_{t_2}, t_2, G))], \quad (1)$$

where $d(\cdot, \cdot)$ is a distance function, e.g., binary cross-entropy. To better leverage the supervision from $\mathbf{x}^*$, an alternative minimizes the distance between each mapping and the ground-truth:

$$\mathcal{L}(\theta) = \mathbb{E}_t [d(f_\theta(\mathbf{x}_t, t, G), \mathbf{x}^*)]. \quad (2)$$

This formulation can be justified by the triangle inequality, which upper-bounds the original objective:

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

This reformulation reduces the problem to supervised learning of the mapping f_θ using a standard loss, such as binary cross-entropy, and allows the model to generalize across different noise levels while consistently recovering the optimal solution.

2.3 Related Work

Neural combinatorial optimization (NCO) has emerged as a promising alternative to traditional heuristics such as LKH [26, 27]. Existing neural solvers can be broadly categorized into autoregressive (AR) and non-autoregressive (NAR) paradigms. An illustrative comparison with our structured diffusion approach is provided in Appendix A, highlighting key differences in solution construction.

AR methods sequentially construct solutions by selecting decision variables step-by-step. Representative works include the Attention Model (AM) [12], which first applied transformer-based architectures to routing problems, and POMO [13], which leverages policy optimization with multiple optima for reinforcement learning. Sym-NCO [14] exploits problem symmetry to enhance generalization capability, while BQ-NCO [15] formulates CO problems under a bisimulation-quotient MDP to improve robustness. The RL4CO repository [16] systematically benchmarks AR-based methods.

NAR methods aim to predict soft-constrained solutions in a one-shot manner, followed by post-processing to ensure feasibility. Early works such as GCN [17] and Att-GCN [18] employ graph neural networks [28, 29, 30] for edge prediction under **supervised learning**. UTSP [20] adopts an **unsupervised learning** framework with scattering attention GNNs, and DIMES [19] proposes a **meta-reinforcement learning** strategy combined with active search. Recently, generative models [31, 32] have shown promise in improving prediction [33], and for CO, **graph diffusion-based generative approaches** [22, 23, 24, 34] model instance-conditioned distributions over high-quality solutions. DIFUSCO [22] achieves SOTA performance on TSP and MIS but lacks instance-specific search. T2T [23] addresses this via an objective-guided gradient search, and Fast T2T [24] further applies consistency training to accelerate diffusion sampling and advance NCO performance. Unify ML4TSP [35] advances a unified modular streamline incorporating existing technologies in both learning and search for both AR and NAR methods.

3 Methodology

We propose a structured diffusion framework for combinatorial optimization. The process begins with a *variable-absorption noising model*, which defines a structure-preserving forward process and its reverse counterpart. To guide the reverse trajectory toward coherent solutions, we introduce a *constrained consistency sampling* strategy that selectively activates high-confidence variables at each step. Finally, we refine the decoded solutions through an *objective-aware, gradient-free method* that integrates task objectives directly into decoding without backward computation. Together, these components form a lightweight and interpretable neural solver.

3.1 Variable-Absorption Noising Process

Our objective is to design a forward diffusion process that generates noisy graph states while strictly respecting combinatorial constraints, e.g. preserving the validity of partial solutions throughout the noising trajectory*. We adopt a discrete diffusion model [22, 23, 24], tailored for structured combinatorial problems where solutions are encoded as binary vectors $\mathbf{x} \in \{0, 1\}^N$. This formulation naturally applies to edge-selection tasks such as TSP and node-selection tasks such as MIS.

It begins with an initial state $\mathbf{x}_0 \sim q(\mathbf{x}_0 \mid G)$ sampled from a data-dependent distribution that yields a feasible solution. Then, a sequence of latent states $\mathbf{x}_{1:T} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$ is generated by progressively injecting noise while maintaining structural validity. The forward process factorizes as $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, where each $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ is a categorical distribution over binary states. Using a one-hot encoding $\tilde{\mathbf{x}} \in \{0, 1\}^{N \times 2}$, where each row denotes the selection status of a variable, the transition is defined as:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \text{Cat}(\mathbf{x}_t; \mathbf{p} = \tilde{\mathbf{x}}_{t-1} \mathbf{Q}_t), \quad q(\mathbf{x}_t \mid \mathbf{x}_0) = \text{Cat}(\mathbf{x}_t; \mathbf{p} = \tilde{\mathbf{x}}_0 \bar{\mathbf{Q}}_t), \quad (4)$$

*Since the process begins from a feasible solution $\mathbf{x}_0$ and injects noise solely by deactivating selected variables, each $\mathbf{x}_t$ along the trajectory is guaranteed to be structurally valid. This guarantee holds only for the forward process; feasibility in the reverse process typically requires post-processing.

where $\mathbf{Q}_t \in [0, 1]^{2 \times 2}$ is the forward transition matrix at time t , and $\overline{\mathbf{Q}}_t = \mathbf{Q}_1 \mathbf{Q}_2 \cdots \mathbf{Q}_t$ is the cumulative transition up to time t . In prior works [22, 23, 24], the transition matrix $\mathbf{Q}_t$ is typically symmetric and doubly stochastic: $\mathbf{Q}_t = \begin{bmatrix} 1 - \beta_t & \beta_t \\ \beta_t & 1 - \beta_t \end{bmatrix}$, where $\beta_t \in [0, 1]$ denotes the transition rate at step t . This formulation enables reversible transitions and yields a uniform stationary distribution, but does not guarantee structural validity of intermediate states.

Variable-Absorption Noising Model. To enforce structural validity at all timesteps, we propose a *variable-absorption noising model* that induces a monotonic decay in variable activations. Specifically, a variable can remain unselected ($\mathbf{x}_i = 0$) or transition from selected ($\mathbf{x}_i = 1$) to unselected, but cannot revert once dropped. This yields a valid absorption trajectory from $\mathbf{x}_0$ to the null solution $\mathbf{x}_T = \mathbf{0}_N \in 0^N$, in which each intermediate state is a feasible partial solution. We implement this behavior using the following asymmetric transition matrix: $\mathbf{Q}_t = \begin{bmatrix} 1 & 0 \\ \beta_t & 1 - \beta_t \end{bmatrix}$, where $\beta_t \in [0, 1]$ denotes the absorption rate at step t . Under this formulation, unselected variables are frozen, and selected variables are independently dropped with probability β_t . This directional and irreversible transition enforces monotonic variable deactivation and ensures that combinatorial constraints are respected throughout the forward trajectory.

3.2 Constrained Consistency Sampling for Reverse Optimization

Following [24, 25], we adopt a multi-step consistency sampling procedure to progressively construct the reverse trajectory of solutions. The process begins from the empty solution $\mathbf{x}_T = \mathbf{0}_N$, and at each timestep τ_n , we aim to generate a more refined intermediate state $\mathbf{x}_{\tau_n}$. Accordingly, we use a consistency model f_θ to predict a clean solution $\mathbf{x}_0$ from the current noisy input: $p_\theta(\mathbf{x}_0 | \mathbf{x}_{\tau_n}, \tau_n, G) = f_\theta(\mathbf{x}_{\tau_n}, \tau_n, G)$, $\mathbf{x}_0 \sim \text{Bernoulli}(p_\theta(\mathbf{x}_0 = 1 | G))$. This predicted $\mathbf{x}_0$ serves as a high-confidence proxy, guiding the sampling of the next intermediate state $\mathbf{x}_{\tau_{n+1}}$ via the variable-absorption process: $\mathbf{x}_{\tau_{n+1}} \sim \text{Cat}(\mathbf{p} = \tilde{\mathbf{x}}_0 \overline{\mathbf{Q}}_{\tau_{n+1}})$, where $\overline{\mathbf{Q}}_\tau$ denotes

Algorithm 1 Multistep Constrained Consistency Sampling

Require: Consistency model $f_\theta(\cdot, \cdot, \cdot)$, graph instance G , time sequence $\tau_1 > \cdots > \tau_{N_\tau - 1}$, threshold δ

- 1: $\mathbf{x}_T = \mathbf{0}_N$
 - 2: $p_\theta(\mathbf{x}_0 | G) \leftarrow f_\theta(\mathbf{x}_T, T, G)$
 - 3: $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0 | G)$
 - 4: **for** $n = 1$ to $N_\tau - 1$ **do**
 - 5: // variable-absorption process
 - 6: $\mathbf{x}_{\tau_n} \sim \text{Cat}(\mathbf{p} = \tilde{\mathbf{x}}_0 \overline{\mathbf{Q}}_{\tau_n})$
 - 7: $p_\theta(\mathbf{x}_0 | G) \leftarrow f_\theta(\mathbf{x}_{\tau_n}, \tau_n, G)$
 - 8: // constrained sampling over $\mathbf{m}$
 - 9: $\mathbf{m} \leftarrow \mathbb{I}(p_\theta(\mathbf{x}_0 = 1 | G) > \delta)$
 - 10: $\mathbf{x}_0 \sim \text{Bernoulli}(\mathbf{m} \odot p_\theta(\mathbf{x}_0 = 1 | G))$
 - 11: **end for**
 - 12: **return** Solution $\mathbf{x}_0$
-

the cumulative absorption transition matrix. The key modeling goal is to construct a trajectory of intermediate states $\mathbf{x}_T, \mathbf{x}_{\tau_1}, \dots, \mathbf{x}_{\tau_{N_\tau - 1}}, \mathbf{x}_0$ that exhibit increasing structure and semantic coherence, ultimately leading to a high-quality solution. This construction process alternates between denoising (via f_θ) and noise injection (via variable-absorption), forming a progressive trajectory over $\mathbf{x}_\tau$. Although each $\mathbf{x}_0$ prediction is discarded after use, it plays a crucial role in steering the generation of the next $\mathbf{x}_\tau$. As illustrated in Fig. 1, this reverse process forms a zigzag pattern between model predictions $\mathbf{x}_0$ and intermediate states $\mathbf{x}_\tau$ with increasing structural coherence, progressively guiding the construction of the final solution. It is important to note that structural coherence across reverse steps is primarily induced by the variable-absorption process, which softly enforces a monotonic selection trajectory ($\mathbf{x}_{\tau_n} \subseteq \mathbf{x}_{\tau_{n+1}}$). For stricter structural guarantees, see our discussion in Appendix B.

Constrained Sampling. To reduce noise and improve the stability of this process, we introduce a *threshold-based constrained sampling strategy*. After obtaining the prediction $p_\theta(\mathbf{x}_0 = 1 | G)$, we suppress low-confidence variables by applying a hard threshold δ :

$$\mathbf{m} = \mathbb{I}(p_\theta(\mathbf{x}_0 = 1 | G) > \delta), \quad \mathbf{x}_0 \sim \text{Bernoulli}(\mathbf{m} \odot p_\theta(\mathbf{x}_0 = 1 | G)). \quad (5)$$

The constrained sampling strategy is also supported by theoretical insights. Combinatorially, it reduces the candidate solution space from 2^N to at most $\sum_{k=0}^{\gamma N} \binom{N}{k}$ for a thresholded fraction $\gamma \in (0, 1)$. Statistically, thresholding the predicted heatmap can be seen as truncating the predictive distribution to a high-confidence region, reducing entropy and sample variance. See proof in Appendix C.

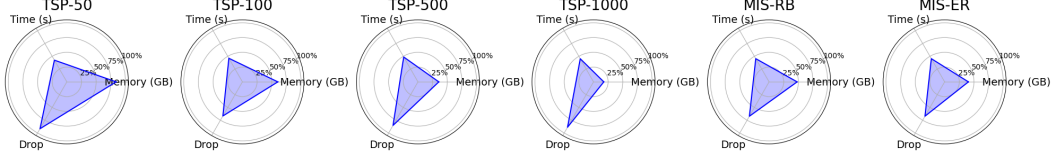


Figure 2: Relative comparison of StruDiCO and Fast T2T across six combinatorial optimization tasks. Metrics include GPU memory usage, inference time, and performance drop (all normalized to Fast T2T = 100%). Lower values indicate improved efficiency or solution quality.

Table 1: Summary of objective-guided refinement framework. StruDiCO enables multi-step *structure construction*, where intermediate states are valid subsets of the soft-constrained solution. B : batch size; L : number of layers; N : number of nodes; d : hidden dimension.

Method	Multi-Step Structure Construction	Objective-Guided			Complexity	
		Gradient-free	Forward	Backward	Time	Memory
Fast T2T [24]	✗	✗	2	1	$\sim O(3BLN^2d)$	$\sim O(2BLN^2d)$
StruDiCO (Ours)	✓	✓	1	0	$O(BLN^2d)$	$O(BLN^2d)$

Theorem 3.1 (Constrained Sampling Reduces Divergence). *Let $q(\mathbf{x}_0 | G)$ denote the true distribution and $p_\theta(\mathbf{x}_0 | \mathbf{x}_\tau, \tau, G)$ the model prediction. Let $\tilde{p}_\theta = \text{Mask}_\delta(p_\theta)$ be the renormalized distribution obtained by thresholding low-confidence variables, $\tilde{p}_\theta(\mathbf{x}_0) = \frac{\mathbb{I}[p_\theta(\mathbf{x}_0) > \delta] p_\theta(\mathbf{x}_0)}{Z_\delta}$, $Z_\delta = \sum_{\mathbf{x}_0} \mathbb{I}[p_\theta(\mathbf{x}_0) > \delta] p_\theta(\mathbf{x}_0)$, where Z_δ is the normalization constant representing the retained probability mass. Then, in terms of the reverse KL divergence, we have $D_{\text{KL}}(\tilde{p}_\theta \| q) \leq D_{\text{KL}}(p_\theta \| q) + \varepsilon_\delta$, where the slack term ε_δ vanishes as the discarded probability mass $1 - Z_\delta \rightarrow 0$.*

Remark. Theorem 3.1 formally shows that threshold-based constrained sampling reduces the divergence between the model and the true solution distributions. Specifically, by masking out low-confidence regions (probability $< \delta$) and renormalizing the remaining mass, the KL divergence decreases whenever the retained region has higher expected likelihood than the discarded region, a condition that is naturally satisfied during inference where high-confidence variables dominate. Importantly, while constrained sampling can also be applied to uniform-noise diffusion models [22, 24], its benefits are substantially amplified under the structure-preserving variable-absorption process, which provides a monotonic and semantically consistent reverse trajectory (see Sec. 4.3).

3.3 Objective-Aware Gradient-Free Refinement

Following the reverse sampling process in Algorithm 1, we finally sample a soft-constrained solution $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0 | G)$. To obtain a feasible solution $\mathbf{s} \in \Omega$, we apply greedy decoding by sequentially inserting edges (for TSP) or nodes (for MIS) with the highest confidence if there are no conflicts, enforcing task-specific constraints. This step ensures hard feasibility and is commonly used in [19, 22, 23, 24]. While feasible, it does not explicitly optimize the task objective and often yields suboptimal solutions. Recent methods like T2T [23] and Fast T2T [24] address this by incorporating task-level supervision via gradient-guided refinement. These methods iteratively perturb a solution and use reverse denoising guided by objective gradients to refine solution,

Algorithm 2 Objective-Aware Gradient-Free Refinement

Require: Consistency model f_θ , graph instance G , prediction p_θ , score function $\text{score}_i = p_i / \phi_i$, number of iterations T_g , perturbation ratio αT

```

1: for  $t = 1$  to  $T_g$  do
2:    $\mathbf{m} \leftarrow \mathbb{I}(p_\theta(\mathbf{x}_0 = 1 | G) > \delta)$ 
3:    $\mathbf{s} \leftarrow \mathbf{m} \odot \mathbf{s}$ 
4:    $\tilde{\mathbf{s}} \leftarrow \text{ONEHOTENCODE}(\mathbf{s})$ 
5:   //apply variable-absorption noise
6:    $\mathbf{s}_{\alpha T} \sim \text{Cat}(\tilde{\mathbf{s}} \cdot \mathbf{Q}_{\alpha T})$ 
7:    $p_\theta \leftarrow f_\theta(\mathbf{s}_{\alpha T} | G)$ 
8:    $\mathbf{s} \leftarrow \text{GREEDYDECODE}(p_\theta, \text{score})$ 
9: end for
10: return  $\mathbf{s}$ 

```

but at the cost of repeated forward-backward passes and reliance on differentiable approximations, making them computationally expensive and less suited to discrete combinatorial settings.

To overcome these practical inefficiencies, we propose a lightweight *objective-aware gradient-free refinement* strategy. Rather than backpropagating gradients, we integrate task objectives directly into

Table 2: Results on TSP-50 and TSP-100. G: Greedy Decoding.

Algorithm	Type	TSP-50			TSP-100		
		Length↓	Drop↓	Time↓	Length↓	Drop↓	Time↓
Concorde [36]	Exact	5.688	0.00%	0.074s	7.756	0.00%	0.404s
LKH3 [26]	Heuristics	5.688	0.00%	0.058s	7.756	0.00%	0.176s
GCN [17]	SL+G+2Opt	5.694	0.115%	0.009s	7.807	0.649%	0.019s
GNNGLS* [37]	SL+G+2Opt	5.707	0.333%	0.019s	7.857	1.295%	0.129s
DIMES* [19]	Meta+RL+G+2Opt	5.823	2.387%	0.018s	8.007	3.232%	0.057s
AM* [12]	RL+G+2Opt	5.679	0.167%	0.048s	7.826	0.898%	0.438s
POMO [13]	RL+G+2Opt	5.693	0.102%	0.019s	7.854	1.253%	0.116s
Sym-NCO [14]	RL+G+2Opt	5.694	0.122%	0.198s	7.818	0.796%	0.634s
BQ-NCO* [15]	RL+G+2Opt	5.795	1.894%	0.205s	7.893	1.772%	0.387s
DIFUSCO ($T_s=50$) [22]	SL+G	5.692	0.076%	0.229s	7.851	1.216%	0.591s
Fast T2T ($T_s=3$) [24]	SL+G	5.694	0.111%	0.024s	7.798	0.537%	0.038s
StruDiCO ($T_s=3$)	SL+G	5.692	0.071%	0.023s	7.786	0.392%	0.037s
T2T ($T_s=50, T_g=30$) [23]	SL+G	5.688	0.015%	0.717s	7.765	0.125%	1.559s
Fast T2T ($T_s=3, T_g=3$) [24]	SL+G	5.688	0.014%	0.139s	7.760	0.052%	0.182s
StruDiCO ($T_s=3, T_g=3$)	SL+G	5.688	0.014%	0.051s	7.758	0.036%	0.082s
DIFUSCO ($T_s=50$) [22]	SL+G+2Opt	5.690	0.046%	0.23s	7.776	0.262%	0.590s
Fast T2T ($T_s=3$) [24]	SL+G+2Opt	5.689	0.031%	0.026s	7.764	0.101%	0.037s
StruDiCO ($T_s=3$)	SL+G+2Opt	5.689	0.019%	0.026s	7.761	0.067%	0.038s
T2T ($T_s=50, T_g=30$) [23]	SL+G+2Opt	5.688	0.012%	1.098s	7.760	0.058%	1.571s
Fast T2T ($T_s=3, T_g=3$) [24]	SL+G+2Opt	5.688	0.012%	0.139s	7.759	0.036%	0.180s
StruDiCO ($T_s=3, T_g=3$)	SL+G+2Opt	5.688	0.011%	0.059s	7.756	0.024%	0.083s

greedy decoding using a objective-aware scoring scheme. Inspired by the relaxed objective redefined in [23], we combine task rewards and constraint penalties to form guidance signals. Specifically, we define the variable-wise score as $\text{score}_i = p_i / (\phi_i + \epsilon)$, where ϵ serves as a balancing term, $p_i = p_\theta(\mathbf{x}_0^i = 1 \mid G)$ denotes the predicted confidence, and ϕ_i encodes task-specific penalties:

$$\phi(i)_{\text{TSP}} = d(i) \quad (\text{edge distance}), \quad \phi(i)_{\text{MIS}} = \sum_{j \in \mathcal{N}(i)} p_j \quad (\text{conflict potential}) \quad (6)$$

This scoring scheme favors confident variables that contribute positively to the task objective, e.g., shorter edges in TSP or nodes likely to form larger independent sets in MIS, thus guiding decoding in an objective-aligned direction without requiring gradient computation.

Gradient-Free Iterative Refinement. Building on the proposed objective-aware scoring scheme, we further introduce an iterative refinement strategy to improve solution quality. Starting from a feasible solution $\mathbf{s}_0$, we apply structure-aware perturbations via variable-absorption noise, denoise the perturbed state using the consistency model, and greedily decode a new solution using the objective-aware score. This process explores a local neighborhood around the current solution and concentrates inference within high-confidence regions. The refinement is fully forward-pass and can be iterated with minimal overhead, as detailed in Algorithm 2. We justify this procedure through the following informal proposition.

Proposition 3.1 (Refinement Enhances Solution Reliability, Informal). *Let $\mathbf{s}$ be a feasible solution decoded from a predictive distribution p_θ using an objective-aware greedy strategy. Assume that the variable-absorption noise operator $\mathbf{Q}_\alpha$ introduces bounded perturbations that preserve the high-confidence regions of p_θ , and that the consistency model f_θ is locally Lipschitz continuous under such perturbations. Then, iterative refinement converges toward fixed points that remain within locally stable, high-confidence regions of the predictive space.*

Remark. Our gradient-free refinement strategy is fundamentally enabled by the structural consistency of the variable-absorption noising model. Since each refinement step perturbs only a subset of already selected variables, the resulting inputs remain semantically aligned with the current solution. This local consistency is preserved by the denoising process and reinforced by objective-aware greedy decoding, enabling stable and meaningful refinement across iterations. In contrast, prior diffusion models based on uniform noise [22, 23, 24] apply symmetric noise to all variables regardless of selection status, which destroys partial solution structure and produces perturbed inputs that are poorly aligned with the original solution. As a result, consistency denoising from such noisy inputs lacks directional guidance, and greedy decoding does not provide reliable improvement. Therefore, while

Table 3: Results on TSP-500 and TSP-1000. S: Sampling Decoding.

Algorithm	Type	TSP-500			TSP-1000		
		Length↓	Drop↓	Time	Length↓	Drop↓	Time
Mathematical Solvers or Heuristics							
Concorde [36]	Exact	16.546	0.00%	18.672s	23.118	0.00%	84.413s
LKH-3 [26]	Heuristics	16.546	0.00%	1.848s	23.119	0.00%	4.641s
Learning-based Solvers with Greedy Decoding							
GCN [17]	SL+G+2OPT	16.899	2.121%	0.128s	—	—	—
DIMES* [19]	RL+G+2Opt	17.165	3.742%	0.453s	—	—	—
BQ-NCO* [15]	RL+G+2Opt	16.838	1.766%	2.454s	23.647	2.287%	5.722s
DIFUSCO ($T_s=50$) [22]	SL+G	18.136	9.611%	1.442s	25.667	11.022%	4.982s
Fast T2T ($T_s=5$) [24]	SL+G	17.467	5.551%	0.251s	24.698	6.831%	0.971s
StruDiCO ($T_s=5$)	SL+G	17.404	5.172%	0.239s	23.118	5.967%	0.900s
T2T ($T_s=50, T_g=30$) [23]	SL+G	17.470	5.578%	3.334s	25.168	8.868%	12.871s
Fast T2T ($T_s=5, T_g=5$) [24]	SL+G	16.919	2.244%	1.426s	23.936	3.539%	5.988s
StruDiCO ($T_s=5, T_g=5$)	SL+G	16.887	2.045%	0.651s	23.755	2.753%	2.533s
DIFUSCO ($T_s=50$) [22]	SL+G+2Opt	16.817	1.641%	1.433s	23.567	1.936%	5.036s
Fast T2T ($T_s=5$) [24]	SL+G+2Opt	16.701	0.922%	0.261s	23.388	1.167%	0.979s
StruDiCO ($T_s=5$)	SL+G+2Opt	16.669	0.728%	0.247s	23.348	0.996%	0.915s
T2T ($T_s=50, T_g=30$) [23]	SL+G+2Opt	16.677	0.793%	3.367s	23.397	1.209%	13.089s
Fast T2T ($T_s=5, T_g=5$) [24]	SL+G+2Opt	16.611	0.383%	1.387s	23.257	0.603%	5.779s
StruDiCO ($T_s=5, T_g=5$)	SL+G+2Opt	16.602	0.326%	0.674s	23.242	0.535%	2.614s
Learning-based Solvers with 4× Sampling Decoding							
DIFUSCO ($T_s=50$) [22]	SL+S	17.533	5.959%	2.131s	25.059	8.399%	19.023s
Fast T2T ($T_s=5$) [24]	SL+S	17.024	2.874%	0.877s	24.096	4.231%	3.635s
StruDiCO ($T_s=5$)	SL+S	16.995	2.696%	0.839s	24.007	3.849%	3.430s
T2T ($T_s=50, T_g=30$) [23]	SL+S	17.054	3.070%	9.722s	24.838	7.444%	34.813s
Fast T2T ($T_s=5, T_g=5$) [24]	SL+S	16.710	0.978%	5.216s	23.712	2.570%	13.346s
StruDiCO ($T_s=5, T_g=5$)	SL+S	16.688	0.846%	2.312s	23.471	1.527%	6.520s
DIFUSCO ($T_s=50$) [22]	SL+S+2Opt	16.694	0.893%	4.941s	23.425	1.326%	19.217s
Fast T2T ($T_s=5$) [24]	SL+S+2Opt	16.633	0.509%	0.911s	23.286	0.726%	3.733s
StruDiCO ($T_s=5$)	SL+S+2Opt	16.611	0.383%	0.857s	23.273	0.670%	3.541s
T2T ($T_s=50, T_g=30$) [23]	SL+S+2Opt	16.621	0.453%	12.636s	23.371	1.070%	36.271s
Fast T2T ($T_s=5, T_g=5$) [24]	SL+S+2Opt	16.580	0.194%	5.095s	23.287	0.419%	13.029s
StruDiCO ($T_s=5, T_g=5$)	SL+S+2Opt	16.575	0.168%	2.437s	23.178	0.261%	6.836s

refinement can in principle also be applied to uniform-noise models, its effectiveness is substantially amplified under the monotonic, structure-preserving property of variable-absorption diffusion.

We summarize the key differences between StruDiCO and prior work [24] in Table 1. Unlike Fast T2T, which incorporates objective guidance during the denoising process, StruDiCO integrates task objectives directly into the decoding stage. This design eliminates the need for backward gradient computation, resulting in significantly lower time and memory complexity while enabling finer control over structure-aware refinement. In addition, Fig. 2 presents a comparative evaluation across six benchmark tasks in terms of GPU memory usage, inference time, and solution quality (Drop), with all metrics normalized to Fast T2T (100%). StruDiCO consistently achieves superior efficiency and performance, with notable improvements on large-scale instances such as TSP-1000 and MIS-ER.

4 Experiment

We evaluate our method on two classic combinatorial optimization tasks—Traveling Salesman Problem (TSP) and Maximum Independent Set (MIS)—across a wide range of graph sizes and difficulty levels. Our evaluation includes comparisons against state-of-the-art (SOTA) exact solvers, heuristics, and neural baselines. For fair comparison, we adopt the standard diffusion-based configuration where T_s denotes the number of initial inference steps and T_g the number of guided refinement steps.

We report three metrics across all settings: (1) **Objective**, the value of the solution (tour *length* for TSP, subset *size* for MIS); (2) **Drop**, the relative performance degradation compared to reference solutions; and (3) **Time**, the average runtime per instance. All models are evaluated with batch size 1 and in single-thread mode unless otherwise specified. More details are provided in the Appendix E.

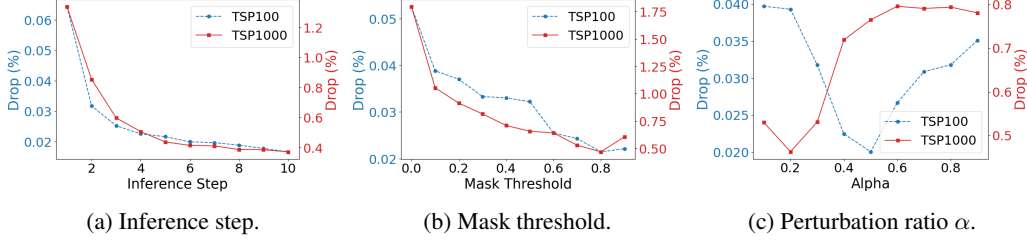


Figure 3: Hyperparameter study on different aspects of our method.

4.1 Results on TSP

Datasets. We follow prior works [12, 22, 23, 24, 35] to generate TSP instances with $N \in \{50, 100, 500, 1000\}$ nodes, uniformly sampled from the unit square. A TSP instance includes N 2-D coordinates and a reference solution obtained by heuristics.

Main Results. As shown in Tables 2 and 3, our method achieves highly competitive solution quality under both greedy and sampling decoding ($\times 4$). For example, on TSP-1000, our method achieves a relative drop of 0.535%, outperforming Fast T2T’s 0.603% under the same $T_s = 5$, $T_g = 5$ configuration. In terms of runtime, our method is $2.2\times$ **faster** on average compared to Fast T2T and over $5\times$ **faster** than T2T. Moreover, Fig. 2 presents radar plots for memory, runtime, and drop across TSP variants, normalized to Fast T2T. Our method consistently reduces GPU memory by **30%–70%**, achieves **40%–65%** **faster** runtime, and improves solution quality, highlighting the benefit of our gradient-free refinement and constrained denoising.

Hyperparameter Study. We conduct hyperparameter studies on TSP-100 and TSP-1000 to analyze the impact of three core hyperparameters in our method: (1) *Inference step and guided step* ($T_s = T_g$). As shown in Fig. 3a, increasing the number of steps improves solution quality up to a point, with performance saturating around $T_s = T_g = 5$ for both instance sizes. (2) *Mask threshold* (δ). Fig. 3b indicates that setting an appropriate threshold δ is critical. Low thresholds retain excessive noise, while overly high values limit exploration. The results demonstrate the tradeoff existence and indicate that $\delta = 0.8$ can achieve relatively better performance. (3) *Perturbation ratio* (α). As shown in Fig. 3c, the refinement quality depends sensitively on the perturbation ratio. Too small values fail to explore alternative candidates, while too large values destroy prior structure. A moderate value ($0.2 \leq \alpha \leq 0.5$) leads to the lowest Drop for TSP-100 and TSP-1000.

As shown in Fig. 3c, the refinement quality depends sensitively on the perturbation ratio. Too small values fail to explore alternative candidates, while too large values destroy prior structure. A moderate value ($0.2 \leq \alpha \leq 0.5$) leads to the lowest Drop for TSP-100 and TSP-1000.

Results for Generalization. We evaluate the cross-scale generalization ability of models by testing on TSP instances of varying sizes, while keeping the training set fixed. Table 4 reports the Drop (%) under greedy decoding. StruDiCO achieves the lowest performance drop in most settings. Notably, when tested on TSP-50, the model trained on TSP-1000 yields a drop of only 0.80%, outperforming Fast T2T (1.26%) and DIFUSCO (2.71%). Similar trends are observed on other test sizes (e.g., TSP-500 or TSP-1K), demonstrating that StruDiCO generalizes well across scales.

4.2 Results on MIS

Datasets. Following prior work [19, 22, 23, 24, 40, 41, 42], we use two types of graphs: random RB graphs [43] with 200-300 nodes and Erdős–Rényi (ER) graphs [44] with 700-800 nodes, where each edges appears independently with probability $p = 0.15$.

Main Results. As shown in Table 5, StruDiCO consistently outperforms diffusion-based baselines in both solution quality and efficiency. On ER graphs, it achieves the best Drop across all decoding

Table 4: Generalization results with *Greedy Decoding*.

Training \ Testing		TSP-50	TSP-100	TSP-500	TSP-1K
TSP-50	DIFUSCO ($T_s=50$)* [22]	0.09%	0.25%	2.55%	2.71%
	T2T ($T_s=50, T_g=30$)* [23]	0.02%	0.11%	1.60%	1.10%
	Fast T2T ($T_s=5, T_g=5$) [24]	0.09%	0.36%	1.02%	1.26%
	StruDiCO ($T_s=5, T_g=5$)	0.01%	0.01%	0.31%	0.80%
TSP-100	DIFUSCO ($T_s=50$)* [22]	1.44%	0.23%	3.44%	3.31%
	T2T ($T_s=50, T_g=30$)* [23]	0.56%	0.17%	2.47%	2.19%
	Fast T2T ($T_s=5, T_g=5$) [24]	0.12%	0.02%	0.40%	0.55%
	StruDiCO ($T_s=5, T_g=5$)	0.16%	0.02%	0.34%	0.55%
TSP-500	DIFUSCO ($T_s=50$)* [22]	4.16%	3.04%	1.40%	1.85%
	T2T ($T_s=50, T_g=30$)* [23]	3.79%	2.25%	0.91%	1.22%
	Fast T2T ($T_s=5, T_g=5$) [24]	2.67%	1.77%	0.38%	0.95%
	StruDiCO ($T_s=5, T_g=5$)	3.46%	2.96%	0.33%	0.49%
TSP-1K	DIFUSCO ($T_s=50$)* [22]	4.54%	3.98%	2.65%	2.21%
	T2T ($T_s=50, T_g=30$)* [23]	4.66%	3.81%	1.61%	1.30%
	Fast T2T ($T_s=5, T_g=5$) [24]	3.46%	3.08%	1.06%	0.58%
	StruDiCO ($T_s=5, T_g=5$)	4.31%	3.22%	0.75%	0.54%

Table 5: Results on MIS. * quoted from [23, 24].

Algorithm	Type	RB-[200-300]			ER-[700-800]		
		SIZE↑	Drop↓	Time	SIZE↑	Drop↓	Time
KaMIS [38]	Heuristics Exact	20.090*	0.00%	45.809s	44.969*	0.00%	60.753s
Gurobi [39]	Exact	20.090	0.00%	22.033s	41.28	8.203%	23.437s
DIFUSCO* ($T_s = 100$) [22]	SL+G	18.52	7.81%	—	36.667	18.462%	4.923s
Fast T2T ($T_s=5$) [24]	SL+G	18.875	6.023%	0.085s	37.820	15.898%	0.262s
StruDiCO ($T_s=5$)	SL+G	19.308	3.899%	0.086s	39.820	11.450%	0.261s
T2T* ($T_s = 50, T_g = 30$) [23]	SL+G	18.98	5.49%	—	39.833	11.419%	6.253s
Fast T2T ($T_s=5, T_g=5$) [24]	SL+G	19.578	2.540%	0.399s	40.781	9.313%	1.217s
StruDiCO ($T_s=5, T_g=5$)	SL+G	19.748	1.707%	0.181s	42.125	6.324%	0.552s
DIFUSCO* ($T_s = 100$) [22]	SL+S	19.13	4.79%	—	39.083	13.092%	17.939s
Fast T2T ($T_s=5$) [24]	SL+S	19.430	3.29%	0.228s	39.767	11.568%	1.0456s
StruDiCO ($T_s=5$)	SL+S	19.756	1.687%	0.229s	41.531	7.645%	0.941s
T2T* ($T_s = 50, T_g = 30$) [23]	SL+S	19.38	3.53%	—	41.417	7.899%	22.426s
Fast T2T ($T_s=5, T_g=5$) [24]	SL+S	19.748	1.715%	1.079s	41.651	7.378%	4.379s
StruDiCO ($T_s=5, T_g=5$)	SL+S	19.908	0.934%	0.492s	43.265	3.789%	1.988s

Table 6: Ablation study on TSP and MIS tasks (Gap, ↓).

Variant	TSP-50	TSP-100	TSP-500	TSP-1000	MIS-RB	MIS-ER
Uniform CM (Fast T2T, $T_s=5$)	0.031%	0.101%	0.922%	1.167%	6.023%	15.898%
VA CM (StruDiCO)	0.032%	0.163%	1.091%	1.183%	7.476%	23.461%
Uniform + CCS	0.025%	0.081%	0.898%	1.074%	5.878%	16.522%
VA + CCS (StruDiCO, $T_s=5$)	0.019%	0.067%	0.728%	0.797%	3.899%	11.450%

modes: **6.324%** under greedy decoding and **3.789%** under sampling, compared to **9.313%** / **7.378%** from Fast T2T and **11.419%** / **7.899%** from T2T. In terms of runtime, StruDiCO is **11×** faster than T2T and **over 2×** faster than Fast T2T under the same refinement setup ($T_s = 5, T_g = 5$).

Generalization. Figures 6 - 9 show generalization performance on ER graphs with varying edge density (p from 0.2 to 0.4) and graph size (from 350-400 to 1400-1600). Our method achieves consistently better solution sizes, outperforming Fast T2T by up to **8%** and DIFUSCO by over **20%** in sparse settings, while maintaining superior scalability.

4.3 Ablation Study

We conduct a detailed ablation study to disentangle the contributions of the Variable-Absorption (VA) mechanism and Constrained Consistency Sampling (CCS). The results are summarized in Table 6. **1)** Replacing the uniform consistency model (CM) with VA slightly degrades performance. This is because the monotonic absorption process ($1 \rightarrow 0$) produces conservative trajectories, structurally coherent but with limited exploration. By contrast, the uniform CM allows symmetric noise injection ($0 \leftrightarrow 1$), which supports broader exploration. **2)** Although CCS improves both uniform and VA models, the gain is substantially larger when combined with VA. This highlights that VA provides semantically consistent intermediate states, enabling CCS to operate in a low-noise, structured space. **Overall**, the ablation confirms that VA and CCS are complementary: VA provides structural continuity, CCS leverages it for reliable variable selection, together accounting for the strong performance.

5 Conclusion

We have introduced a structured denoising diffusion framework tailored for CO problems. By modeling the forward process as a variable absorption process and the reverse process as a progressive variable selection mechanism, it enables the construction of interpretable partial solutions at each inference step. We also propose a gradient-free objective-aware decoding strategy that explicitly incorporates objective constraints to enhance inference quality on unseen instances. Extensive experiments on TSP and MIS show the superior efficiency and effectiveness, achieving up to $3.5\times$ faster inference, 70% lower memory usage, and substantial improvements in solution quality.

References

- [1] Léo Baty, Kai Jungel, Patrick S Klein, Axel Parmentier, and Maximilian Schiffer. Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. *Transportation Science*, 58(4):708–725, 2024.
- [2] Guman Singh and Mohammad Rizwanullah. Combinatorial optimization of supply chain networks: A retrospective & literature review. *Materials today: proceedings*, 62:1636–1642, 2022.
- [3] Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Mingyuan Ma, Zhaoyang Shen, Juejian Wu, Yuanfan Xu, Hengrui Zhang, Kai Zhong, et al. Machine learning for electronic design automation: A survey. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 26(5):1–46, 2021.
- [4] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d’horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.
- [5] Michele Lombardi and Michela Milano. Boosting combinatorial problem modeling with machine learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5472–5478, 2018.
- [6] Andoni I. Garmendia, Josu Ceberio, and Alexander Mendiburu. Applicability of neural combinatorial optimization: a critical view. *ACM Transactions on Evolutionary Learning and Optimization*, 4(3):1–26, 2024.
- [7] Wenzheng Pan, Hao Xiong, Jiale Ma, Wentao Zhao, Yang Li, and Junchi Yan. Unico: On unified combinatorial optimization via problem reduction to matrix-encoded general tsp. In *International Conference on Learning Representations*, 2025.
- [8] Haoyu Geng, Hang Ruan, Runzhong Wang, Yang Li, Yang Wang, Lei Chen, and Junchi Yan. Benchmarking pto and pno methods in the predictive combinatorial optimization regime. *Advances in Neural Information Processing Systems*, 37:65944–65971, 2024.
- [9] Yang Li, Xinyan Chen, Wenxuan Guo, Xijun Li, Wanqian Luo, Junhua Huang, Hui-Ling Zhen, Mingxuan Yuan, and Junchi Yan. Hardsatgen: Understanding the difficulty of hard sat formula generation and a strong structure-hardness-aware baseline. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4414–4425, 2023.
- [10] Xinyan Chen, Yang Li, Runzhong Wang, and Junchi Yan. Mixsatgen: Learning graph mixing for sat instance generation. In *The Twelfth International Conference on Learning Representations*, 2024.
- [11] Ziao Guo, Yang Li, Chang Liu, Wenli Ouyang, and Junchi Yan. Acm-milp: Adaptive constraint modification via grouping and selection for hardness-preserving milp instance generation. In *Forty-first International Conference on Machine Learning*, 2024.
- [12] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.
- [13] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21188–21198, 2020.
- [14] Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 35, pages 1936–1949, 2022.
- [15] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pages 77416–77429, 2023.

- [16] Federico Berto, Chuanbo Hua, Junyoung Park, Laurin Luttmann, Yining Ma, Fanchen Bu, Jiarui Wang, Haoran Ye, Minsu Kim, Sanghyeok Choi, et al. RL4co: an extensive reinforcement learning for combinatorial optimization benchmark. *arXiv preprint arXiv:2306.17100*, 2023.
- [17] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.
- [18] Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7474–7482, 2021.
- [19] Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 35, pages 25531–25546, 2022.
- [20] Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. In *Advances in Neural Information Processing Systems*, volume 36, pages 47264–47278, 2023.
- [21] Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *International Conference on Machine Learning*, pages 43346–43367, 2024.
- [22] Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. In *Advances in neural information processing systems*, volume 36, pages 3706–3731, 2023.
- [23] Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pages 50020–50040, 2023.
- [24] Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 37, pages 30179–30206, 2024.
- [25] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023.
- [26] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12:966–980, 2017.
- [27] Shipai Zhou, Yuandong Ding, Chi Zhang, Zhiguang Cao, and Yan Jin. Dualopt: A dual divide-and-optimize algorithm for the large-scale traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27178–27186, 2025.
- [28] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.
- [29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.
- [30] Yu Wang, Liang Hu, Xiaofeng Cao, Yi Chang, and Ivor W Tsang. Enhancing locally adaptive smoothing of graph neural networks via laplacian node disagreement. *IEEE Transactions on Knowledge and Data Engineering*, 36(3):1099–1112, 2023.
- [31] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [32] Yang Li, Yichuan Mo, Liangliang Shi, and Junchi Yan. Improving generative adversarial networks via adversarial learning in latent space. *Advances in neural information processing systems*, 35:8868–8881, 2022.

- [33] Yang Li, Jiale Ma, Yebin Yang, Qitian Wu, Hongyuan Zha, and Junchi Yan. Generative modeling reinvents supervised learning: Label repurposing with predictive consistency learning. In *Forty-second International Conference on Machine Learning*.
- [34] André Hottung, Bhanu Bhandari, and Kevin Tierney. Learning a latent search space for routing problems using variational autoencoders. In *International Conference on Learning Representations*, 2021.
- [35] Yang Li, Jiale Ma, Wenzheng Pan, Runzhong Wang, Haoyu Geng, Nianzu Yang, and Junchi 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.
- [36] David Applegate, Ribert Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver. <https://www.math.uwaterloo.ca/tsp/concorde/index.html>, 2006. Accessed: 2025-04-21.
- [37] Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. In *International Conference on Learning Representations*, 2022.
- [38] Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F Werneck. Finding near-optimal independent sets at scale. In *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)*, pages 138–150. SIAM, 2016.
- [39] Gurobi Optimization. Gurobi optimizer reference manual. <http://www.gurobi.com>, 2020. Accessed: 2025-04-21.
- [40] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in neural information processing systems*, volume 31, 2018.
- [41] Maximilian Böther, Otto Kißig, Martin Taraz, Sarel Cohen, Karen Seidel, and Tobias Friedrich. What’s wrong with deep learning in tree search for combinatorial optimization. In *International Conference on Learning Representations*, 2022.
- [42] Sungsoo Ahn, Younggyo Seo, and Jinwoo Shin. Learning what to defer for maximum independent sets. In *International Conference on Machine Learning*, pages 134–144. PMLR, 2020.
- [43] Dinghui Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with gflownets. In *Advances in neural information processing systems*, volume 36, pages 11952–11969, 2023.
- [44] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.
- [45] Christian Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & operations research*, 31(12):1985–2002, 2004.
- [46] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.
- [47] Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. Coexpander: Adaptive solution expansion for combinatorial optimization. In *Forty-second International Conference on Machine Learning*. PMLR, 2025.

A Discussion on the Construction Paradigms of NCO

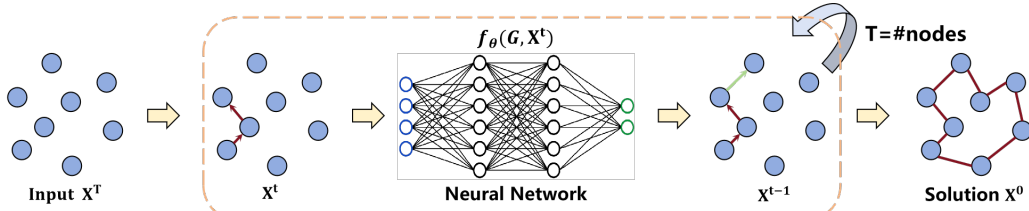
As illustrated in Fig. 4, **autoregressive** (AR) methods construct solutions by selecting decision variables (nodes or edges) in a strictly sequential manner, typically requiring $O(N)$ decoding steps, where N is the number of decision variables. In contrast, **non-autoregressive** (NAR) methods predict soft-constrained solutions in a one-shot pass, followed by heuristic post-processing to ensure feasibility, effectively reducing decision-making to a single step. These two paradigms represent the two extremes of variable selection granularity in neural combinatorial optimization (NCO).

Naturally, this raises the question: *Can we design a construction paradigm that balances granularity and efficiency, lying between these two extremes?* Recently, generative approaches based on diffusion models have achieved state-of-the-art performance and offer a promising direction. However, they suffer from two critical limitations:

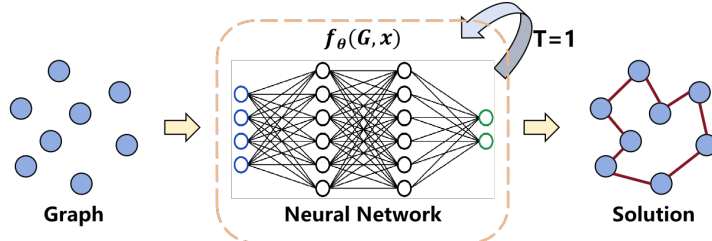
1. **Excessive inference steps.** Traditional diffusion models typically involve hundreds or thousands of steps (e.g., $T = 1000$), often exceeding the number of variables in the problem, which introduces significant inefficiency.
2. **Lack of structural interpretability in intermediate states.** As shown in Fig. 5a, intermediate solutions during denoising are usually dense and noisy (e.g., full graphs for TSP), offering little insight into how variables are progressively determined.

The recent introduction of **consistency models** [25] has mitigated the first issue by enabling high-quality solutions with as few as 3-5 inference steps, referred as FastT2T [24]. Building on this progress, our work addresses the second issue by proposing a **structured diffusion framework** with step-wise interpretability.

As shown in Fig. 5b, our model begins from an empty solution and progressively activates decision variables with high predicted confidence at each inference step. This process can be viewed as a variable selection trajectory, where each intermediate state incrementally extends the previous one, forming a structurally consistent sequence of partial solutions.



(a) The solving pipeline of Autoregressive construction.



(b) The solving pipeline of Non-Autoregressive construction.

Figure 4: Comparison of solving pipelines between Autoregressive and Non-Autoregressive constructions.

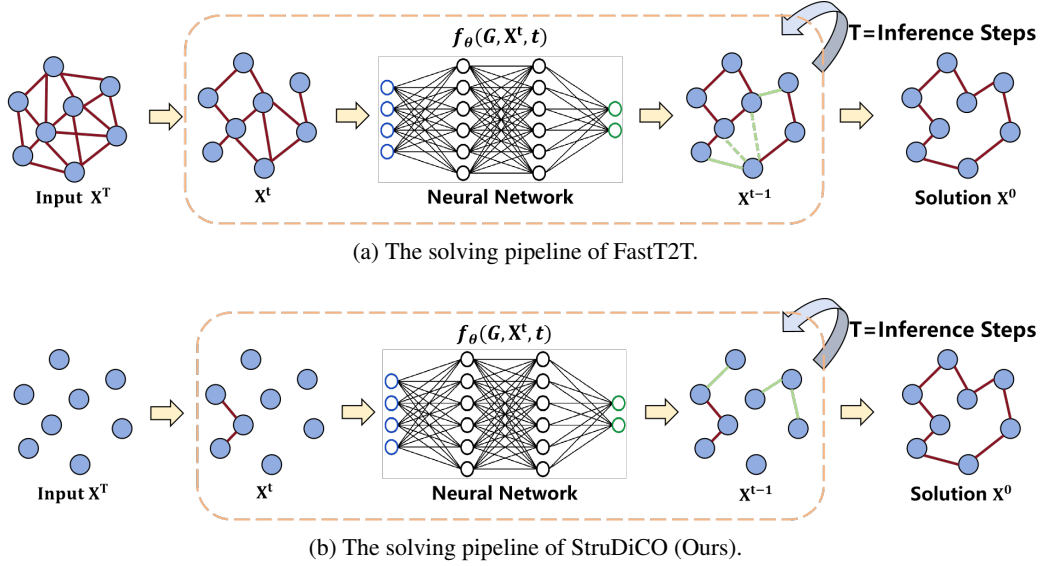


Figure 5: Comparison of solving pipelines between FastT2T and StruDiCO. Green solid lines indicate newly added edges, while green dashed lines represent deleted edges.

B Discussion on Structural Consistency in Denoising

B.1 Limitations of Vanilla Reverse Sampling

Most diffusion-based CO solvers, including our main pipeline (see Sec. 3.2), follow the standard multistep consistency sampling protocol [24, 25], as shown in Algorithm 3, which iteratively denoises and resamples $\mathbf{x}_0$ given intermediate states $\mathbf{x}_\tau$. While this process can recover high-quality solutions, it does not strictly enforce any structural constraints on the reverse trajectory, such as **feasibility preservation** across steps.

Algorithm 3 Multistep Consistency Sampling

Require: Consistency model $f_\theta(\cdot, \cdot, \cdot)$, graph instance G , time sequence $\tau_1 > \dots > \tau_{N_\tau-1}$

- 1: Sample $\mathbf{x}_T \sim \mathcal{U}$
 - 2: $p_\theta(\mathbf{x}_0 | G) \leftarrow f_\theta(\mathbf{x}_T, T, G)$
 - 3: $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0 | G)$
 - 4: **for** $n = 1$ to $N_\tau - 1$ **do**
 - 5: $\mathbf{x}_{\tau_n} \sim \text{Cat}(\mathbf{p} = \tilde{\mathbf{x}}_0 \overline{\mathbf{Q}}_{\tau_n})$
 - 6: $p_\theta(\mathbf{x}_0 | G) \leftarrow f_\theta(\mathbf{x}_{\tau_n}, \tau_n, G)$
 - 7: $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0 | G)$
 - 8: **end for**
 - 9: **return** Solution $\mathbf{x}_0$
-

B.2 Step-Wise Decoding Sampling for Intermediate Feasibility

To further improve step-wise feasibility, we explore a stronger variant: **step-wise greedy decoding**. At each reverse step τ , we rank the predicted heatmap $p_\theta(\mathbf{x}_0 = 1 | \mathbf{x}_\tau, \tau, G)$ in descending order and activate the top n_τ variables to form $\mathbf{x}_\tau$, subject to task-specific constraints (e.g., no subtours or conflicts). Here, n_τ is determined by the expected number of active variables given by the cumulative transition matrix $\overline{\mathbf{Q}}_\tau$ under the reverse schedule.

This approach better aligns the reverse process with the semantics of the variable-absorption forward process and can be combined with feasibility filters (e.g., constraint checks) to construct strictly valid $\mathbf{x}_\tau$ at every step. However, systematic exploration of this strategy is left as future work.

C Full Proof of Theorem 3.1

We provide the complete mathematical proof of Theorem 3.1, which formally analyzes the effect of threshold-based masking on the KL divergence between the predicted and true solution distributions.

Theorem C.1 (Constrained Sampling Reduces Divergence, Full Version). *Let $q(\mathbf{x}_0 \mid G)$ be the true distribution and $p_\theta(\mathbf{x}_0 \mid \mathbf{x}_\tau, \tau, G)$ the model prediction. Fix a threshold $\delta \in (0, 1)$ and define the constrained support*

$$\mathcal{M}_\delta = \{i \in [N] : p_\theta(\mathbf{x}_0^i = 1 \mid \mathbf{x}_\tau, \tau, G) > \delta\}, \quad (7)$$

$$\mathcal{S}_\delta = \{\mathbf{x}_0 : \text{supp}(\mathbf{x}_0) \subseteq \mathcal{M}_\delta\}. \quad (8)$$

Let the masked (renormalized) model be

$$\tilde{p}_\theta(\mathbf{x}_0) = \frac{p_\theta(\mathbf{x}_0) \mathbb{I}[\mathbf{x}_0 \in \mathcal{S}_\delta]}{Z_\delta}, \quad (9)$$

$$Z_\delta = \sum_{\mathbf{x}_0 \in \mathcal{S}_\delta} p_\theta(\mathbf{x}_0) \in (0, 1], \quad (10)$$

and define the log-likelihood ratio $\ell(\mathbf{x}_0) = \log \frac{p_\theta(\mathbf{x}_0)}{q(\mathbf{x}_0)}$. Assume $q(\mathbf{x}_0) > 0$ whenever $p_\theta(\mathbf{x}_0) > 0$ (so that $D_{\text{KL}}(p_\theta \| q) < \infty$). Then the reverse-KL difference admits the exact decomposition

$$D_{\text{KL}}(\tilde{p}_\theta \| q) = D_{\text{KL}}(p_\theta \| q) - (1 - Z_\delta) (\Delta_\delta + c_\delta), \quad (11)$$

where

$$\Delta_\delta = \mathbb{E}_{p_\theta(\cdot | \mathcal{S}_\delta)}[\ell(\mathbf{x}_0)] - \mathbb{E}_{p_\theta(\cdot | \mathcal{S}_\delta)}[\ell(\mathbf{x}_0)], \quad (12)$$

$$c_\delta = \frac{-\log Z_\delta}{1 - Z_\delta} (\geq 0). \quad (13)$$

Consequently,

$$D_{\text{KL}}(\tilde{p}_\theta \| q) \leq D_{\text{KL}}(p_\theta \| q) + \varepsilon_\delta, \quad (14)$$

$$\varepsilon_\delta = (1 - Z_\delta) [-(\Delta_\delta + c_\delta)]_+, \quad (15)$$

and in particular, if $\Delta_\delta \geq -c_\delta$ then $D_{\text{KL}}(\tilde{p}_\theta \| q) \leq D_{\text{KL}}(p_\theta \| q)$, with strict inequality whenever $Z_\delta < 1$ and the inequality is strict.

Proof. All sums below are over the discrete space of $\mathbf{x}_0$. We use the shorthand $p(\mathbf{x}_0) = p_\theta(\mathbf{x}_0 \mid \mathbf{x}_\tau, \tau, G)$, $q(\mathbf{x}_0) = q(\mathbf{x}_0 \mid G)$, and $\tilde{p}(\mathbf{x}_0) = \tilde{p}_\theta(\mathbf{x}_0)$, suppressing the conditioning for readability.

Step 1: Reverse-KL under masking. By definition of $\tilde{p}$ and Z_δ ,

$$\tilde{p}(\mathbf{x}_0) = \begin{cases} \frac{p(\mathbf{x}_0)}{Z_\delta}, & \mathbf{x}_0 \in \mathcal{S}_\delta, \\ 0, & \mathbf{x}_0 \notin \mathcal{S}_\delta. \end{cases} \quad (16)$$

Since $q > 0$ wherever $p > 0$, we have $q > 0$ on $\mathcal{S}_\delta$ (because $p > 0$ there if and only if $\tilde{p} > 0$). Then

$$D_{\text{KL}}(\tilde{p} \| q) = \sum_{\mathbf{x}_0 \in \mathcal{S}_\delta} \tilde{p}(\mathbf{x}_0) \log \frac{\tilde{p}(\mathbf{x}_0)}{q(\mathbf{x}_0)} \quad (17)$$

$$= \sum_{\mathbf{x}_0 \in \mathcal{S}_\delta} \frac{p(\mathbf{x}_0)}{Z_\delta} \log \frac{p(\mathbf{x}_0)/Z_\delta}{q(\mathbf{x}_0)} \quad (18)$$

$$= \sum_{\mathbf{x}_0 \in \mathcal{S}_\delta} \frac{p(\mathbf{x}_0)}{Z_\delta} \log \frac{p(\mathbf{x}_0)}{q(\mathbf{x}_0)} - \log Z_\delta \sum_{\mathbf{x}_0 \in \mathcal{S}_\delta} \frac{p(\mathbf{x}_0)}{Z_\delta} \quad (19)$$

$$= \mathbb{E}_{p(\cdot | \mathcal{S}_\delta)} \left[\log \frac{p(\mathbf{x}_0)}{q(\mathbf{x}_0)} \right] - \log Z_\delta \quad (20)$$

$$= \mathbb{E}_{p(\cdot | \mathcal{S}_\delta)} [\ell(\mathbf{x}_0)] - \log Z_\delta. \quad (21)$$

Step 2: Reverse-KL of the original model. Decompose p into its mass on $\mathcal{S}_\delta$ and $\mathcal{S}_\delta^c$:

$$D_{\text{KL}}(p\|q) = \sum_{\mathbf{x}_0} p(\mathbf{x}_0) \log \frac{p(\mathbf{x}_0)}{q(\mathbf{x}_0)} \quad (22)$$

$$= \sum_{\mathbf{x}_0 \in \mathcal{S}_\delta} p(\mathbf{x}_0) \ell(\mathbf{x}_0) + \sum_{\mathbf{x}_0 \notin \mathcal{S}_\delta} p(\mathbf{x}_0) \ell(\mathbf{x}_0) \quad (23)$$

$$= Z_\delta \mathbb{E}_{p(\cdot|\mathcal{S}_\delta)}[\ell(\mathbf{x}_0)] + (1 - Z_\delta) \mathbb{E}_{p(\cdot|\mathcal{S}_\delta^c)}[\ell(\mathbf{x}_0)]. \quad (24)$$

Step 3: Taking the difference. Subtracting (B) from (A) gives

$$D_{\text{KL}}(\tilde{p}\|q) - D_{\text{KL}}(p\|q) \quad (25)$$

$$= \left(\mathbb{E}_{p(\cdot|\mathcal{S}_\delta)}[\ell] - \log Z_\delta \right) - \left(Z_\delta \mathbb{E}_{p(\cdot|\mathcal{S}_\delta)}[\ell] + (1 - Z_\delta) \mathbb{E}_{p(\cdot|\mathcal{S}_\delta^c)}[\ell] \right) \quad (26)$$

$$= (1 - Z_\delta) \mathbb{E}_{p(\cdot|\mathcal{S}_\delta)}[\ell] - (1 - Z_\delta) \mathbb{E}_{p(\cdot|\mathcal{S}_\delta^c)}[\ell] - \log Z_\delta \quad (27)$$

$$= -(1 - Z_\delta) \underbrace{\left(\mathbb{E}_{p(\cdot|\mathcal{S}_\delta^c)}[\ell] - \mathbb{E}_{p(\cdot|\mathcal{S}_\delta)}[\ell] \right)}_{\Delta_\delta} - \log Z_\delta \quad (28)$$

$$= -(1 - Z_\delta) \Delta_\delta - \log Z_\delta. \quad (29)$$

When $Z_\delta = 1$ the mask is identity and both sides are equal; for $Z_\delta \in (0, 1)$ rewrite the last term as

$$-\log Z_\delta = (1 - Z_\delta) \frac{-\log Z_\delta}{1 - Z_\delta} = (1 - Z_\delta) c_\delta, \quad (30)$$

which yields the exact decomposition Eq. (11):

$$D_{\text{KL}}(\tilde{p}\|q) = D_{\text{KL}}(p\|q) - (1 - Z_\delta) (\Delta_\delta + c_\delta). \quad (31)$$

Step 4: Upper bound and sufficient condition. Since for any real x , $x = x_+ - (-x)_+$ with $x_+ = \max\{x, 0\}$, we get the one-sided bound

$$D_{\text{KL}}(\tilde{p}\|q) \leq D_{\text{KL}}(p\|q) + (1 - Z_\delta) [- (\Delta_\delta + c_\delta)]_+. \quad (32)$$

Moreover, whenever $\Delta_\delta \geq -c_\delta$ (equivalently $\mathbb{E}_{p(\cdot|\mathcal{S}_\delta^c)}[\ell] \geq \mathbb{E}_{p(\cdot|\mathcal{S}_\delta)}[\ell] - c_\delta$), the difference in Eq. (11) is nonpositive and thus $D_{\text{KL}}(\tilde{p}\|q) \leq D_{\text{KL}}(p\|q)$, with strict improvement if $Z_\delta < 1$ and the inequality is strict. $\square$

D Supplementary Experiments

D.1 Applied to Capacitated Vehicle Routing Problem

To further demonstrate the generality of StruDiCO beyond classical edge-selection (TSP) and node-selection (MIS) tasks, we extend it to the Capacitated Vehicle Routing Problem (CVRP), a more challenging CO task that involves both global routing structures and hard feasibility constraints, such as customer demands and vehicle capacities. The refinement procedure for CVRP follows the same design as that for TSP: greedy decoding is performed based on the learned heatmap, with the objective guided by a distance-based penalty term. We evaluate StruDiCO on CVRP-50, CVRP-100, and CVRP-200 under standard benchmark settings [16]. To further improve both solution feasibility and cost, we incorporate a local search heuristic (Classic-LS [45]) during inference. As summarized in Table 7, these results demonstrate that StruDiCO effectively generalizes to CVRP, consistently improving over learning-based baselines while remaining efficient.

D.2 Results on Real-World Data

We evaluate our model trained on randomly generated 100-node instances against real-world TSPLIB benchmarks containing 50-200 nodes. For fair comparison, we adopt the same hyperparameter settings as [24], with $T_s = T_g = 10$. As shown in Table 8, StruDiCO demonstrates superior performance, achieving an average optimality gap of only 0.13%.

Table 7: Results on CVRP. LS: Local Search.

Algorithm	Type	CVRP 50			CVRP 100			CVRP 200		
		Length↓	Drop↓	Time	Length↓	Drop↓	Time	Length↓	Drop↓	Time
HGS [46]	Heuristics	10.37	0.00%	1s	15.56	0.00%	20s	19.63	0.00%	60s
Sym-NCO [16]	RL+LS	10.57	1.95%	0.09s	15.93	2.37%	0.19s	20.19	2.86%	0.36s
COExpander [47]	SL+LS	10.77	3.90%	0.04s	16.22	4.25%	0.06s	20.59	4.89%	0.15s
StruDiCO ($T_s=3, T_g=3$)	SL+LS	10.48	1.12%	0.05s	15.85	1.88%	0.11s	20.25	3.19%	0.26s
StruDiCO ($T_s=5, T_g=5$)	SL+LS	10.45	0.85%	0.07s	15.80	1.53%	0.17s	20.16	2.71%	0.38s

Table 8: Solution quality for methods trained on random 100-node problems and evaluated on **TSPLIB instances with 50-200 nodes**. * denotes results quoted from previous works [24].

Instances	AM*	GCN*	Learn2OPT*	GNNGLS*	DIFUSCO*	T2T*	Fast T2T*	Ours
eil51	16.767%	40.025%	1.725%	1.529%	2.82%	0.14%	0.00%	0.00%
berlin52	4.169%	33.225%	0.449%	0.142%	0.00%	0.00%	0.00%	0.00%
st70	1.737%	24.785%	0.040%	0.764%	0.00%	0.00%	0.00%	0.00%
eil76	1.992%	27.411%	0.096%	0.163%	0.34%	0.00%	0.00%	0.00%
pr76	0.816%	27.793%	1.228%	0.039%	1.12%	0.40%	0.00%	-0.00%
rat99	2.645%	17.633%	0.123%	0.550%	0.09%	0.09%	0.00%	0.00%
kroA100	4.017%	28.828%	18.313%	0.728%	0.10%	0.00%	0.00%	0.00%
kroB100	5.142%	34.686%	1.119%	0.147%	2.29%	0.74%	0.65%	0.00%
kroC100	0.972%	35.506%	0.349%	1.571%	0.00%	0.00%	0.00%	0.00%
kroD100	2.717%	38.018%	0.866%	0.572%	0.07%	0.00%	0.00%	0.00%
kroE100	1.470%	26.589%	1.832%	1.216%	3.83%	0.27%	0.13%	2.15%
rd100	3.407%	50.432%	1.725%	0.003%	0.08%	0.00%	0.00%	0.11%
eil101	2.994%	21.776%	1.529%	0.03%	0.03%	0.00%	0.00%	0.00%
lin105	1.739%	34.902%	1.867%	0.606%	0.00%	0.00%	0.00%	0.18%
pr107	3.933%	80.564%	0.898%	0.439%	0.91%	0.61%	0.62%	0.18%
pr124	2.677%	70.146%	10.232%	0.755%	1.02%	0.08%	0.08%	-0.00%
bier127	5.908%	45.561%	3.044%	1.948%	0.94%	0.54%	1.50%	0.04%
ch130	3.182%	39.090%	0.709%	3.519%	0.29%	0.06%	0.00%	0.24%
pr136	5.064%	58.673%	0.000%	3.387%	0.19%	0.10%	0.01%	0.04%
pr144	7.641%	55.837%	1.526%	3.581%	0.80%	0.50%	0.39%	0.00%
ch150	1.584%	49.743%	0.321%	2.113%	0.57%	0.49%	0.00%	0.04%
kroA150	3.784%	45.411%	0.724%	2.984%	0.34%	0.14%	0.00%	0.24%
kroB150	2.437%	56.745%	0.886%	3.258%	0.30%	0.00%	0.07%	0.00%
pr152	7.494%	49.376%	3.119%	3.119%	1.69%	0.83%	0.19%	0.69%
u159	7.551%	38.338%	0.054%	1.020%	0.82%	0.00%	0.00%	0.00%
rat195	6.893%	24.968%	0.743%	1.666%	1.48%	1.27%	0.79%	-0.00%
d198	373.020%	62.952%	0.522%	4.727%	3.32%	1.97%	0.86%	-0.00%
kroA200	7.106%	40.885%	1.441%	2.029%	2.28%	0.57%	0.49%	0.00%
kroB200	8.541%	43.643%	0.646%	2.589%	2.35%	0.92%	2.50%	-0.00%
Mean	16.767%	40.025%	1.725%	1.529%	0.97%	0.35%	0.28%	0.13%

D.3 Supplementary Results of Generalization Study

To further evaluate the generalization ability of StruDiCO, we conduct additional experiments on MIS instances with varying edge densities and graph sizes. Specifically, we test models trained on ER graphs with $p = 0.15$ and $n = 700$ – 800 on out-of-distribution graphs with different p values (0.2, 0.3, 0.4) and node counts (350–400, 1400–1600). As shown in Figures 6–9, StruDiCO consistently outperforms prior diffusion-based baselines (e.g., DIFUSCO [22], T2T [23], Fast T2T [24]) under both greedy and sampling decoding, demonstrating superior adaptability to distribution shifts in both graph sparsity and scale.

E Experimentnal Details

E.1 Hardware

All models are trained and tested using NVIDIA A40 (48G) GPUs and Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz.

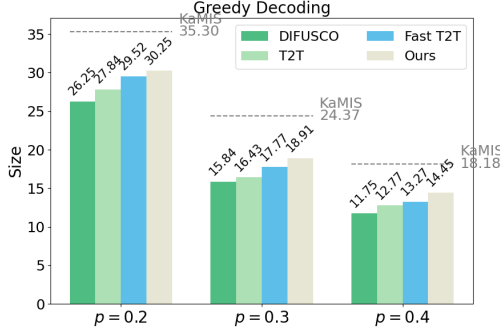


Figure 6: Generalization performance from $p = 0.15$ to $p = 0.2$, $p = 0.3$, and $p = 0.4$.

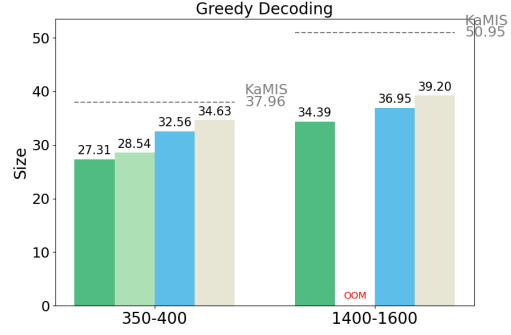


Figure 7: Generalization performance from ER 700–800 to ER 350–400 and 1400–1600.

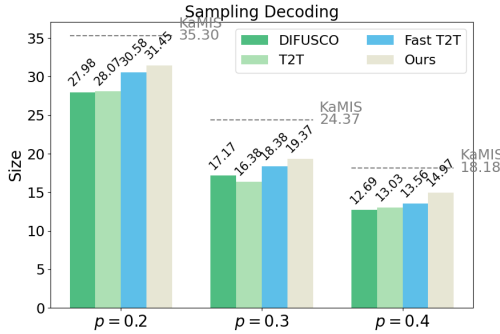


Figure 8: Generalization performance from $p = 0.15$ to $p = 0.2$, $p = 0.3$, and $p = 0.4$.

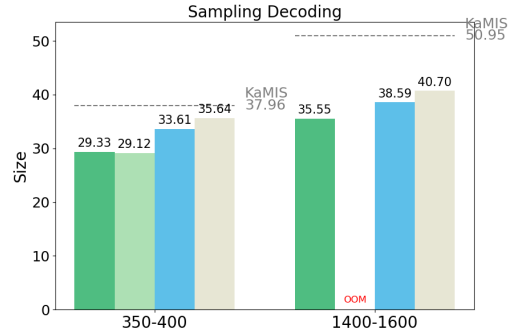


Figure 9: Generalization performance from ER 700–800 to ER 350–400 and 1400–1600.

E.2 Sparsification

For large-scale TSP problems, we follow [22, 23, 24] to employ sparse graphs, as sparsified by constraining each node to connect to only its k nearest neighbors, determined by Euclidean distances. For TSP-500, we set $k = 50$, and for TSP-1000, $k = 100$. This strategy prevents the exponential increase in edges typical in dense graphs as node count rises.

E.3 Dataset

Our dataset settings follow previous works [22, 23, 24] to ensure fair and consistent comparisons. The reference solutions for TSP-50/100 are labeled using the Concorde exact solver [36], while TSP-500/1000 solutions are generated by the LKH-3 heuristic solver [26]. The test sets for TSP-50/100 contain 1280 instances sourced from [12, 17], and those for TSP-500/1000 contain 128 instances from [18]. For the MIS task, we use KaMIS [38] to generate reference solutions for both RB and ER graphs. The RB training set consists of 90,000 randomly generated instances, and the test set includes 500 instances. For ER graphs, we generate 163,840 training instances, and the test set is obtained from [19].

E.4 Hyperparameters

We have organized the training settings and model parameters of StruDiCO in Table 9.

Table 9: Details about the training hyperparameters of StruDiCO.

Problem	Dataset Size	Batch Size	Epoch	Learning Rate	Hidden Dimension	Noise Degree
TSP-50	1,502k	32	50	2e-4	256	0.5
TSP-100	1,502k	12	50	2e-4	256	0.5
TSP-500	128k	6	50	2e-4	256	0.5
TSP-1000	64k	4	50	2e-4	256	0.5
MIS-RB	90k	4	50	2e-4	256	0.3
MIS-ER	163k	4	50	2e-4	128	0.5

F Network Architecture Details

F.1 Input Embedding Layer

Given node vector $x \in \mathbb{R}^{N \times 2}$, weighted edge vector $e \in \mathbb{R}^E$, denoising timestep $t \in \{\tau_1, \dots, \tau_M\}$, where N denotes the number of nodes in the graph, and E denotes the number of edges, we compute the sinusoidal features of each input element respectively:

$$\tilde{x}_i = \text{concat}(\tilde{x}_{i,0}, \tilde{x}_{i,1}) \quad (21)$$

$$\tilde{x}_{i,j} = \text{concat} \left(\sin \frac{x_{i,j}}{T^{0/d}}, \cos \frac{x_{i,j}}{T^{0/d}}, \sin \frac{x_{i,j}}{T^{2/d}}, \cos \frac{x_{i,j}}{T^{2/d}}, \dots, \sin \frac{x_{i,j}}{T^{d/d}}, \cos \frac{x_{i,j}}{T^{d/d}} \right) \quad (22)$$

$$\tilde{e}_i = \text{concat} \left(\sin \frac{e_i}{T^{0/d}}, \cos \frac{e_i}{T^{0/d}}, \sin \frac{e_i}{T^{2/d}}, \cos \frac{e_i}{T^{2/d}}, \dots, \sin \frac{e_i}{T^{d/d}}, \cos \frac{e_i}{T^{d/d}} \right) \quad (23)$$

$$\tilde{t} = \text{concat} \left(\sin \frac{t}{T^{0/d}}, \cos \frac{t}{T^{0/d}}, \sin \frac{t}{T^{2/d}}, \cos \frac{t}{T^{2/d}}, \dots, \sin \frac{t}{T^{d/d}}, \cos \frac{t}{T^{d/d}} \right) \quad (24)$$

where d is the embedding dimension, T is a large number (usually selected as 10000), and $\text{concat}(\cdot)$ denotes concatenation.

Next, we compute the input features of the graph convolution layer:

$$x_i^0 = W_1^0 \tilde{x}_i \quad (25)$$

$$e_i^0 = W_2^0 \tilde{e}_i \quad (26)$$

$$t^0 = W_4^0 (\text{ReLU}(W_3^0 \tilde{t})) \quad (27)$$

where $t^0 \in \mathbb{R}^{d_t}$, d_t is the time feature embedding dimension. Specifically, for TSP, the embedding input edge vector e is a weighted adjacency matrix, which represents the distance between different nodes, and e^0 is computed as above. For MIS, we initialize e^0 to a zero matrix $0^{E \times d}$.

F.2 Graph Convolution Layer

Following [17], the cross-layer convolution operation is formulated as:

$$x_i^{l+1} = x_i^l + \text{ReLU}(\text{BN}(W_1^l x_i^l + \sum_{j \sim i} \eta_{ij}^l \odot W_2^l x_j^l)) \quad (28)$$

$$e_{ij}^{l+1} = e_{ij}^l + \text{ReLU}(\text{BN}(W_3^l e_{ij}^l + W_4^l x_i^l + W_5^l x_j^l)) \quad (29)$$

$$\eta_{ij}^l = \frac{\sigma(e_{ij}^l)}{\sum_{j' \sim i} \sigma(e_{ij'}^l) + \epsilon} \quad (30)$$

where x_i^l and e_{ij}^l denote the node feature vector and edge feature vector at layer l , $W_1, \dots, W_5 \in \mathbb{R}^{h \times h}$ denote the model weights, and η_{ij}^l denotes the dense attention map. The convolution operation integrates the edge feature to accommodate the significance of edges in routing problems.

For TSP, we aggregate the timestep feature with the edge convolutional feature and reformulate the update for edge features as follows:

$$e_{ij}^{l+1} = e_{ij}^l + \text{ReLU}(\text{BN}(W_3^l e_{ij}^l + W_4^l x_i^l + W_5^l x_j^l)) + W_6^l(\text{ReLU}(t^0)) \quad (31)$$

For MIS, we aggregate the timestep feature with the node convolutional feature and reformulate the update for node features as follows:

$$x_i^{l+1} = x_i^l + \text{ReLU}(\text{BN}(W_1^l x_i^l + \sum_{j \sim i} \eta_{ij}^l \odot W_2^l x_j^l)) + W_6^l(\text{ReLU}(t^0)) \quad (32)$$

F.3 Output Layer

The prediction of the edge heatmap in TSP and node heatmap in MIS is as follows:

$$e_{i,j} = \text{Softmax}(\text{norm}(\text{ReLU}(W_e e_{i,j}^L))) \quad (33)$$

$$x_i = \text{Softmax}(\text{norm}(\text{ReLU}(W_n x_i^L))) \quad (34)$$

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

F.4 Hyper-parameters

For both TSP and MIS tasks, we construct a 12-layer GCN derived above. We set the node, edge, and timestep embedding dimension $d = 256, 128$ for TSP and MIS tasks, respectively.

G Limitations and Broader Impacts

Despite the effectiveness of our proposed method, several limitations remain. First, our model is trained under a supervised learning paradigm, which relies on access to a substantial number of labeled optimal or near-optimal solutions. While this setting is practical for many standard CO benchmarks, it may present limitations when extending to domains where high-quality supervision is difficult or expensive to obtain. Second, although we demonstrate improved generalization over existing diffusion-based solvers [24] in fixed-scale settings, the model still faces challenges in handling real-world CO problems with unknown or variable instance sizes. Third, while our diffusion process incrementally constructs solutions in a structured manner, it still follows a non-autoregressive paradigm and thus requires heuristic post-processing (e.g., greedy decoding) to enforce final feasibility. This dependency may constrain its applicability in tasks where domain-specific heuristics are unavailable or expensive to design.

This work contributes to the growing field of learning-based combinatorial optimization, with potential applications in logistics, circuit design, scheduling, and operations research. By introducing interpretable and efficient generative methods, our approach can improve decision-making in domains that require both scalability and transparency.

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: Sec. 1 and Sec. 4.

Guidelines:

- The answer NA 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 NA 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: Appendix G

Guidelines:

- The answer NA 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: [\[Yes\]](#)

Justification: Appendix C

Guidelines:

- The answer NA 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: Appendix E

Guidelines:

- The answer NA 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: [No]

Justification: The source code will be made publicly available upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.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://nips.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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Appendix E

Guidelines:

- The answer NA 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: [No]

Justification: We follow the setting of previous works [22, 23, 24] to report the average solution quality over 128 or 1,280 instances in Sec. 4.

Guidelines:

- The answer NA 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: Appendix E

Guidelines:

- The answer NA 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 NA 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: Appendix G

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: no such risks

Guidelines:

- The answer NA 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 have been cited.

Guidelines:

- The answer NA 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: [NA]

Justification: Currently, the paper does not release new assets. Our source code will be released upon the acceptance of the paper with comprehensive documents.

Guidelines:

- The answer NA 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: [NA]

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

Guidelines:

- The answer NA 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: [NA]

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

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA 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 (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.