DRIFTCO: A DRIFTING MODEL-BASED SOLVER FOR COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion model-based solvers have recently achieved strong performance in neural combinatorial optimization (CO). They generate candidates through stochastic denoising trajectories, which can provide both strong solution quality and useful candidate diversity. This makes them well suited to best-of-many pipelines, where multiple candidates are locally improved and the best solution is retained. However, multi-step denoising introduces substantial per-candidate inference cost. Recent acceleration methods distill the trajectory into fewer steps or even a single step, but such distillation can weaken the candidate diversity needed for effective best-of-many search. To mitigate this issue, we propose DriftCO, a drifting model-based one-step CO solver that promotes candidate quality and diversity through a training-time attraction–repulsion mechanism, rather than through long inference-time denoising trajectories. This mechanism encourages the model to favor high-quality solution regions while avoiding collapse into redundant search neighborhoods, allowing DriftCO to generate diverse candidates with strong solution quality in one step at inference time. Experiments on the traveling salesman problem and maximum independent set across multiple problem sizes demonstrate the effectiveness of DriftCO.

1 INTRODUCTION

Combinatorial optimization (CO) problems aim to find optimal solutions over discrete decision spaces and underlie a wide range of applications, including routing, scheduling, and graph optimization (Drakulic et al., 2025; Chen et al., 2026; Gao et al., 2026; Zhang et al., 2025a). CO problems are typically NP-hard (Weissman et al., 2025; Wang et al., 2026a; Bi et al., 2026; Kim et al., 2024; Guo et al., 2026), making efficient solution methods challenging to design. As a result, traditional approaches have largely relied on carefully designed heuristics and handcrafted solvers (Garmendia et al., 2026; Reijnen et al., 2025; Soler & Raidl, 2025; Gu et al., 2025; Verdù et al., 2025). Recently, learning-based neural CO solvers have emerged as a promising paradigm (Wang et al., 2024; Feng et al., 2026; Zhou et al., 2024a; Bu et al., 2024; Surendran et al., 2026). Rather than serving as standalone replacements for classical heuristics, various successful methods generate multiple candidate solutions, refine them with local search, and then retain the best one (Berto et al., 2025b; Pan et al., 2025; Xiao et al., 2024a; Jin et al., 2024; Luo et al., 2026). This best-of-many pipeline benefits from candidates that combine strong solution quality with sufficient diversity across search neighborhoods. Specifically, candidates should approach high-quality solution regions while covering complementary neighborhoods rather than collapsing to redundant ones.

Building on this best-of-many pipeline, diffusion model (DM)-based methods (Sun & Yang, 2023; Li et al., 2023; Zhao et al., 2025) have recently emerged as a powerful class of neural CO solvers. These methods learn a stochastic denoising process that transforms random noise into candidate solutions through iterative inference-time trajectories. This generation process naturally supports best-of-many solving: Iterative denoising progressively moves noisy states toward high-quality solution regions, while stochastic transitions accumulated along multi-step trajectories promote coverage of different search neighborhoods. After local improvement and best-of-many selection, this combination increases the chance that at least one candidate reaches a favorable region.

Despite enabling DM-based solvers to generate high-quality and diverse candidates, multi-step denoising trajectories introduce substantial per-candidate inference cost (Sun & Yang, 2023; Li et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2023; Zhao et al., 2025). Recent works therefore seek to accelerate DM-based CO solving by distilling the denoising trajectory into fewer steps or even a single step (Li et al., 2024; Wang et al., 2025b). However, as empirically illustrated in Section 5.3, such distillation can weaken the diversity induced by multi-step stochastic denoising. Consequently, maintaining diverse candidates may require additional perturbation, resampling, or test-time search, which reintroduces extra inference-time computation and can reduce the practical benefit of trajectory distillation. These observations suggest that one-step generative CO solving requires more than trajectory distillation: It should preserve the candidate diversity needed for effective best-of-many search while maintaining the efficiency of single-step generation.

Motivated by these observations, we propose DriftCO, to our knowledge the first drifting model-based solver for CO (Deng et al., 2026). DriftCO learns one-step candidate generation through a training-time attraction–repulsion mechanism: Attraction steers candidates toward high-quality solution regions, while repulsion discourages them from collapsing into redundant neighborhoods. Importantly, DriftCO requires only a single reference solution for each training instance. Single-reference training is particularly well suited to CO for both practical and task-specific reasons. Obtaining multiple distinct high-quality solutions for every instance can require costly repeated optimization (Sanokowski et al., 2024). Moreover, multiple positive samples are important in the original image-generation setting since training aims to capture variation in the target data distribution rather than concentrate on a single image (Deng et al., 2026). In CO, by contrast, the goal for each instance is to obtain a high-quality solution rather than to recover the full set of optimal solutions. When multiple optima exist, any one of them attains the same optimal objective value. Accordingly, a single high-quality reference can serve as a quality anchor, while diversity is promoted through interactions among generated candidates. This enables DriftCO to train directly on standard single-reference CO datasets without constructing additional solution annotations. At inference time, each candidate is generated in a single forward pass without denoising trajectories. Experiments on the traveling salesman problem (TSP) and maximum independent set (MIS) across multiple sizes demonstrate the effectiveness and efficiency of DriftCO.

2 Related Work

We review related work on learning-based neural CO solvers by distinguishing whether the solution generation process is based on diffusion models. Accordingly, we divide existing methods into two categories: Non-DM-based neural CO solvers and DM-based neural CO solvers. We further discuss drifting models (Deng et al., 2026), whose training-time generative mechanism motivates DriftCO to learn direct candidate generation that favors promising solution regions while avoiding redundant search neighborhoods.

Non-DM-based neural CO solvers. Learning-based neural CO solvers that do not use DMs can be broadly categorized by how they produce solutions. Improvement-based methods (Xu et al., 2026; Ye et al., 2023; Li et al., 2025a; Chen & Tian, 2019; Xin et al., 2021; Lu et al., 2019; Ma et al., 2023; Kim et al., 2025; Garmendia, 2026) start from an initial solution and iteratively refine it. Construction-based methods (Bello et al., 2017; Vinyals et al., 2015; Khalil et al., 2017; Kool et al., 2019; Liao et al., 2025; Kwon et al., 2020; Drakulic et al., 2023; Kim et al., 2022b) directly generate a solution from the input instance, while hybrid methods (Kim et al., 2022a; Zhou et al., 2025; Ye et al., 2024; Kong et al., 2024; Zhang et al., 2025b; Wang et al., 2026c) combine elements of both paradigms. Within construction-based solvers, autoregressive methods (Zhang et al., 2023; Zhang & Cao, 2025; Liu et al., 2026; Luo et al., 2025b; Jiang et al., 2025; Son et al., 2024; Hoppe et al., 2025; Luo et al., 2025c;a; Gui et al., 2026; Jin et al., 2023; Gao et al., 2023; Goh et al., 2024) generate solutions sequentially, whereas non-autoregressive methods (Fu et al., 2021; Qiu et al., 2022; Joshi et al., 2019; Karalias et al., 2025; Min et al., 2023; Xia et al., 2024; Xiao et al., 2024b) predict structured decision representations that are decoded into feasible solutions.

DM-based CO solvers. Starting from DIFUSCO (Sun & Yang, 2023), this paradigm has evolved along several directions. One direction revisits the generative process itself, with recent works proposing continuous-time and structure-aware diffusion variants (Seo, 2025; Wang et al., 2025a; Zhao et al., 2025; Chen & Han, 2026). Another direction aims to improve test-time inference through trajectory distillation, gradient-guided sampling, or search-augmented generation (Huang

et al., 2023; Li et al., 2023; 2024; 2025b; Lei et al., 2026; Basson & Preux, 2024; Wang et al., 2025b). A separate line of work explores alternative training paradigms beyond supervised imitation, such as reinforcement learning or unsupervised energy-based objectives (Sanokowski et al., 2024; 2025; Song et al., 2026; Hong et al., 2024). Other approaches propose new solving paradigms, including adaptive expansion strategies and flow-based generative formulations, to further improve efficiency, stability, and solution quality (Ma et al., 2025a; Wang et al., 2026b; Li et al., 2026; Friedmann & Dym, 2026). Among these developments, the works most closely related to ours are those that reduce, restructure, or bypass long inference-time denoising trajectories. One line of work compresses iterative diffusion through progressive distillation, consistency training, or self-consistency objectives, enabling few-step or one-step generation (Huang et al., 2023; Li et al., 2024; Wang et al., 2025a;b). Another line redesigns the generative process itself through analytically simplified diffusion, discrete flow matching, or deterministic geometric transport (Zhao et al., 2025; Li et al., 2026; Friedmann & Dym, 2026). Recent methods also improve solution quality through adaptive expansion or inference-time search and refinement, often iteratively updating partial or perturbed solutions during generation (Ma et al., 2025a; Li et al., 2025b; Wang et al., 2026b). In contrast, DriftCO approaches efficient candidate generation from the training side. Rather than compressing or restructuring an inference-time trajectory, DriftCO uses drifting model-based attraction–repulsion learning to train a direct generator that favors high-quality solution regions while discouraging redundant candidate neighborhoods. After training, each candidate is generated with a single forward pass, without inference-time denoising trajectories or search loops.

Drifting models (Deng et al., 2026) are a recent one-step generative modeling paradigm. They train a pushforward generator with a drifting field, enabling samples to be generated directly from prior noise without iterative inference-time trajectories. Recent work further connects drifting models to DMs through smoothed score matching (Lai et al., 2026). This connection motivates our use of DM-based methods as the main comparison class for DriftCO.

While drifting models provide the generative principle behind DriftCO, applying this paradigm to graph-based CO requires adapting the framework to the structure of combinatorial solutions. This leads to three design choices in DriftCO. 1) Standard drifting models generate continuous data samples directly, whereas CO solutions are discrete and the objective is generally non-differentiable with respect to the generated solution. We therefore let the model generate continuous heatmaps as differentiable surrogates of discrete decisions, and obtain feasible solutions through decoding. This heatmap-based formulation provides an interface between continuous generative modeling and discrete CO solving. 2) DriftCO reformulates the drifting field in an instance-wise conditional form. Original drifting models define the drift over a shared continuous data space, where samples can be compared through a common similarity kernel. In graph-based CO, however, a solution heatmap is tied to the input graph: Its entries index decision variables whose semantics are instance-specific, such as edge decisions in TSP or node decisions in MIS. Therefore, heatmaps from different instances should not be mixed when estimating attraction and repulsion, even if they have the same dimensionality. To make the drift well-defined, we introduce the conditional pushforward distribution $q_\theta(\cdot \mid G)$ and estimate the drifting field within each graph instance. This ensures that attraction and repulsion are computed only among representations defined over the same node or edge decisions. 3) DriftCO redesigns the similarity kernel in the drifting field to better capture structural redundancy among generated candidates. Original drifting models typically use Euclidean-distance-based kernels, which are suitable for continuous data but less aligned with combinatorial decision structures. In graph-based CO, two heatmaps may differ substantially in coordinate-wise magnitude while still emphasizing similar node or edge decisions. Using Euclidean distance can therefore provide a poor measure of whether generated candidates occupy redundant solution neighborhoods. We instead introduce a structural cosine kernel that measures the alignment of decision-confidence patterns. This allows repulsion to act more strongly between structurally similar candidates.

3 PRELIMINARIES

3.1 COMBINATORIAL OPTIMIZATION ON GRAPHS

A broad class of CO problems can be formulated on graphs (Zheng et al., 2024; Berto et al., 2025c; Rydin et al., 2026; Chen et al., 2025; Berto et al., 2025a). Specifically, let $G = (V, E)$ denote an input graph with node set V and edge set E. The goal is to find a feasible solution $\mathbf{x} \in \mathcal{X}(\mathcal{G})$ that

optimizes a task-specific objective function $f(\mathbf{x}, \mathcal{G})$, where $\mathcal{X}(\mathcal{G})$ denotes the feasible solution space associated with graph $\mathcal{G}$. A common formulation represents solutions as binary decision vectors $\mathbf{x} \in \{0,1\}^D$, where D denotes the number of decision variables. Each dimension corresponds to selecting or rejecting a node or an edge in the graph. In TSP (Fang et al., 2024; Hottung et al., 2025a; Bi et al., 2024), the input is a complete graph $G = (V, E)$ with edge weights representing pairwise distances between cities. The objective is to find a minimum-cost Hamiltonian tour (Fan et al., 2025; Zhou et al., 2024b; Chalumeau et al., 2025) that visits each city exactly once. In MIS (Ma et al., 2025b; Li et al., 2025c; Ahn et al., 2020), given an undirected graph $G = (V, E)$, the goal is to find the largest subset of vertices $S \subseteq V$ such that no two vertices in S are adjacent.

3.2 Diffusion Model-Based Combinatorial Optimization Solvers

DM-based CO solvers formulate solution generation as a conditional denoising process over a problem instance G. Let $\mathbf{x}_0 \in \{0,1\}^D$ denote a discrete solution representation, such as selected edges or nodes. A forward diffusion process gradually corrupts $\mathbf{x}_0$ into noisy latent states $\mathbf{x}_1, \ldots, \mathbf{x}_T$ according to a prescribed noise schedule, such that $\mathbf{x}_T$ approaches a tractable noise distribution. A neural model is then trained to reverse this process conditioned on G. Starting from $\mathbf{x}_T \sim p(\mathbf{x}_T)$, the reverse process defines an inference-time trajectory

$$p_\theta(\mathbf{x}_{0:T-1} \mid \mathbf{x}_T, G) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, G),$$

where each transition progressively refines the current noisy candidate toward a structured solution representation. The factorization above highlights that DM-based CO solvers obtain candidate generation through inference-time denoising trajectories. Learned reverse transitions progressively refine noisy states toward structured solution representations, thereby improving candidate quality. Meanwhile, different initial noise samples and stochastic transitions accumulated along the trajectory provide diversity across candidates. Thus, in DM-based CO solvers, both candidate quality and diversity are closely tied to the iterative inference-time trajectory.

4 DriftCO

We propose DriftCO, a drifting model-based CO solver that directly maps a problem graph and noise vector to a continuous solution heatmap, providing a relaxation of discrete node or edge decisions. During training, an instance-wise drifting field uses a single reference solution for quality guidance and same-instance repulsion to discourage redundant candidates. At inference time, each candidate is generated in a single forward pass and decoded into a feasible solution.

4.1 Graph-Conditioned Pushforward at Training Time

For each problem instance G, let $\mathbf{y}_G^* \in \{0,1\}^D$ denote its single high-quality reference solution, where each dimension corresponds to a task-specific decision variable. For edge-decision problems such as TSP, $D = |E|$, while for node-decision problems such as MIS, $D = |V|$. To enable gradient-based learning over discrete decisions, we adopt a continuous relaxation of the solution space (Sun & Yang, 2023). Specifically, the generator outputs a solution heatmap $\hat{\mathbf{x}} \in \mathbb{R}^D$, whose entries serve as soft confidence scores for decoding a feasible discrete solution. In CO, the semantics and quality of a solution are inherently tied to the underlying problem instance. We therefore model solution generation conditionally on the input graph rather than through a single unconditional distribution shared across instances. Given a problem graph G and a noise vector $\mathbf{z} \sim p_{\mathbf{z}}$, e.g., a standard Gaussian, a neural generator f_θ produces a continuous solution heatmap $\hat{\mathbf{x}} = f_\theta(\mathbf{z}, G) \in \mathbb{R}^D$.

For a fixed graph G, the graph-conditioned generator $f_\theta(\cdot, G)$ maps the noise prior $p_{\mathbf{z}}$ to a conditional distribution over solution heatmaps. We denote the resulting distribution by $q_\theta(\cdot \mid G)$. Formally, $q_\theta(\cdot \mid G)$ is the pushforward distribution (Deng et al., 2026) of $p_{\mathbf{z}}$ under $f_\theta(\cdot, G)$:

$$q_\theta(\cdot \mid G) = \big(f_\theta(\cdot, G)\big)_{\#} p_{\mathbf{z}}. \tag{1}$$

Our goal is to learn a conditional generator that places probability mass in high-quality solution regions while retaining useful variation across generated candidates.

Since neural network training is iterative, the optimization process produces a sequence of generators $\{f_{\theta_i}\}$ and corresponding conditional pushforward distributions $\{q_{\theta_i}(\cdot \mid G)\}$. In DriftCO, the evolution of these distributions is governed by two complementary signals: The single reference solution $\mathbf{y}_G^*$ provides instance-specific quality guidance, while interactions among generated candidates discourage concentration in redundant solution neighborhoods. When the model parameters are updated from θ_i to θ_{i+1}, a generated heatmap undergoes an implicit drift (Deng et al., 2026):

$$\hat{\mathbf{x}}_{i+1} = \hat{\mathbf{x}}_i + \Delta\hat{\mathbf{x}}_i, \qquad \Delta\hat{\mathbf{x}}_i := f_{\theta_{i+1}}(\mathbf{z}, G) - f_{\theta_i}(\mathbf{z}, G). \tag{2}$$

The residual $\Delta\hat{\mathbf{x}}_i$ describes the displacement of a generated heatmap induced by the parameter update. To explicitly govern this drift, we next construct an instance-wise conditional drifting field that combines attraction toward the single reference solution with repulsion among generated candidates.

4.2 Instance-wise Conditional Drifting Field

In graph-based CO, decision variables have instance-specific semantics, as each variable corresponds to a node or edge defined by the underlying graph (Goh et al., 2025; Hottung et al., 2025b; Ouyang et al., 2026). Thus, the identity of each decision variable is tied to the topology of its graph G. Under this graph-binding property, evaluating similarities between solutions from different graphs is semantically ill-defined, even when the instances share the same dimensionality. We therefore compute both attraction and repulsion within each graph instance: The reference solution provides an instance-specific quality anchor, while generated candidates interact only with other candidates defined over the same decision variables.

Designing the instance-wise field. For each instance G, let $\mathbf{y}_G^*$ denote its single high-quality reference solution. In the original drifting formulation, attraction is defined as a kernel-normalized mean over positive samples (Deng et al., 2026). Under our single-reference supervision, the positive distribution reduces to a point mass at $\mathbf{y}_G^*$, and the normalized attraction therefore simplifies exactly to $\mathbf{y}_G^* - \hat{\mathbf{x}}$. Based on this simplification, we define the attraction field as

$$\mathbf{V}^+(\hat{\mathbf{x}}; G) := \mathbf{y}_G^* - \hat{\mathbf{x}}. \tag{3}$$

The same-instance repulsion field is defined over other generated samples from the conditional distribution $q_\theta(\cdot \mid G)$:

$$\mathbf{V}_{q_\theta}^-(\hat{\mathbf{x}}; G) := \frac{1}{Z_{q_\theta}(\hat{\mathbf{x}})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\theta(\cdot|G)} \big[\kappa(\hat{\mathbf{x}}, \tilde{\mathbf{x}})(\tilde{\mathbf{x}} - \hat{\mathbf{x}})\big]. \tag{4}$$

where $\kappa(\cdot, \cdot)$ is a similarity kernel and $Z_{q_\theta}(\hat{\mathbf{x}}) := \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\theta(\cdot|G)} \big[\kappa(\hat{\mathbf{x}}, \tilde{\mathbf{x}})\big]$. The attraction field pulls the generated heatmap toward the single reference solution, while the repulsion field points toward regions occupied by other generated candidates for the same instance. Following the attraction–repulsion principle of drifting models (Deng et al., 2026), we define the total instance-wise drifting field by subtracting repulsion from attraction:

$$\mathbf{V}(\hat{\mathbf{x}}; G) := \mathbf{V}^+(\hat{\mathbf{x}}; G) - \mathbf{V}_{q_\theta}^-(\hat{\mathbf{x}}; G) = \frac{1}{Z_{q_\theta}(\hat{\mathbf{x}})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\theta(\cdot|G)} \big[\kappa(\hat{\mathbf{x}}, \tilde{\mathbf{x}})(\mathbf{y}_G^* - \tilde{\mathbf{x}})\big]. \tag{5}$$

This decomposition separates the two roles of the drifting field. The single reference solution anchors candidate quality, whereas repulsion among generated samples discourages the conditional generator from concentrating all probability mass in redundant solution neighborhoods. Candidate diversity is therefore induced by interactions among the generated candidates rather than supplied through multiple reference solutions.

To absorb this desired drift into the generator parameters, we follow the drift-regression formulation (Deng et al., 2026). For a generated heatmap $f_\theta(\mathbf{z}, G)$, its drifted target is obtained by adding the instance-wise field and stopping gradients through the target:

$$\mathcal{L}(\theta) = \mathbb{E}_{G \sim \mathcal{D}} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left\| f_\theta(\mathbf{z}, G) - \mathrm{sg}\big(f_\theta(\mathbf{z}, G) + \mathbf{V}(f_\theta(\mathbf{z}, G); G)\big) \right\|_2^2, \tag{6}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. The resulting parameter update incorporates quality anchoring and diversity-promoting interactions entirely during training, enabling direct candidate generation without inference-time denoising trajectories.

4.3 Structural Cosine Kernel

The repulsion term in Eq. 5 uses a similarity kernel $\kappa(\cdot,\cdot)$ to determine the interaction strength between generated candidates. The original drifting model (Deng et al., 2026) adopts a Laplacian kernel based on Euclidean distance. However, Euclidean distance is sensitive to coordinate-wise magnitude differences and does not directly reflect whether two heatmaps emphasize similar combinatorial decisions. We therefore use a structural cosine kernel:

$$\kappa(\hat{\mathbf{x}},\tilde{\mathbf{x}}) = \exp\Big(\frac{\cos(\hat{\mathbf{x}},\tilde{\mathbf{x}})-1}{\tau}\Big), \tag{7}$$

where $\cos(\hat{\mathbf{x}},\tilde{\mathbf{x}}) = (\hat{\mathbf{x}}\cdot\tilde{\mathbf{x}})/(\|\hat{\mathbf{x}}\|_2\|\tilde{\mathbf{x}}\|_2)$ and $\tau>0$ is a temperature hyperparameter. Cosine similarity emphasizes the alignment of confidence patterns rather than their absolute scale, assigning larger interaction weights to generated heatmaps that favor similar decision variables. Consequently, repulsion acts more strongly among structurally redundant candidates and less strongly between candidates that already explore different solution neighborhoods.

This interpretation is closely connected to overlap between discrete combinatorial solutions. A valid binary solution $\mathbf{y}\in\{0,1\}^D$ represents the set of selected decision variables $\mathcal{S}_{\mathbf{y}}$, such as selected nodes or edges. When both inputs are binary, cosine similarity reduces to the Ochiai coefficient (Chowdhury, 2010):

$$\cos(\mathbf{y}',\mathbf{y}) = \frac{\mathbf{y}'\cdot\mathbf{y}}{\|\mathbf{y}'\|_2\|\mathbf{y}\|_2} = \frac{|\mathcal{S}_{\mathbf{y}'}\cap\mathcal{S}_{\mathbf{y}}|}{\sqrt{|\mathcal{S}_{\mathbf{y}'}|\cdot|\mathcal{S}_{\mathbf{y}}|}}. \tag{8}$$

Thus, as relaxed heatmaps approach structured decision patterns, the cosine kernel naturally reflects overlap in the selected combinatorial variables. This allows the drifting field to focus its diversity-promoting repulsion on candidates occupying similar structural neighborhoods rather than merely separating them according to raw Euclidean distance.

4.4 Training and Inference

Training objective. We estimate the instance-wise drifting field with samples generated during stochastic training (Deng et al., 2026). At each training step, we sample a batch of B graph instances $\{G_b\}_{b=1}^{B}$, where each instance G_b is paired with a single high-quality reference solution $\mathbf{y}_b^*$. For each instance, we draw K independent noise vectors $\{\mathbf{z}_b^{(k)}\}_{k=1}^{K}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ and generate relaxed heatmaps $\hat{\mathbf{x}}_b^{(k)} = f_\theta(\mathbf{z}_b^{(k)},G_b)$. The reference solution determines the attraction term, while the K generated heatmaps provide empirical samples from $q_\theta(\cdot\mid G_b)$ for estimating same-instance repulsion. For the k-th generated heatmap, we estimate the drifting field as

$$\widehat{\mathbf{V}}_b^{(k)} = \mathbf{y}_b^* - \hat{\mathbf{x}}_b^{(k)} - \frac{\sum_{j\neq k}\kappa(\hat{\mathbf{x}}_b^{(k)},\hat{\mathbf{x}}_b^{(j)})(\hat{\mathbf{x}}_b^{(j)}-\hat{\mathbf{x}}_b^{(k)})}{\sum_{j\neq k}\kappa(\hat{\mathbf{x}}_b^{(k)},\hat{\mathbf{x}}_b^{(j)})}. \tag{9}$$

Thus, attraction is determined by the single reference solution, whereas repulsion is estimated entirely from the set of generated candidates for the same graph. No interactions are computed across different graph instances. The model is trained by regressing each generated heatmap toward its stop-gradient drifted target:

$$\mathcal{L}(\theta) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{K}\sum_{k=1}^{K}\Big\|\hat{\mathbf{x}}_b^{(k)} - \mathrm{sg}\Big(\hat{\mathbf{x}}_b^{(k)}+\widehat{\mathbf{V}}_b^{(k)}\Big)\Big\|_2^2. \tag{10}$$

This objective implements an empirical drift-regression update: The estimated drifting field specifies a displacement for each generated heatmap, while the stop-gradient operator treats the corresponding drifted target as fixed when updating the generator. The attraction term anchors candidate quality, whereas kernel-weighted same-instance repulsion discourages different noise samples from concentrating in redundant solution neighborhoods. The complete training procedure is summarized in Algorithm 1 in Appendix B.

Heatmap generation. At inference time, DriftCO generates each candidate with a single forward pass. Given a problem instance G, we sample $\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ and compute $\hat{\mathbf{x}} = f_\theta(\mathbf{z},G)$. The resulting heatmap provides confidence scores over the decision variables and is converted into a feasible solution through task-specific decoding. Multiple candidates are obtained by drawing independent noise vectors, without inference-time denoising or iterative refinement.

5 EXPERIMENTS

In this section, we evaluate DriftCO on TSP and MIS, focusing on inference efficiency and zero-shot generalization across problem scales and graph distributions.

5.1 EXPERIMENTS FOR TSP

Datasets. We conduct experiments on Euclidean TSP instances with N two-dimensional coordinates. Following DIFUSCO (Sun & Yang, 2023), we generate synthetic instances by uniformly sampling N nodes from the unit square $[0, 1]^2$. For TSP-50 and TSP-100, we train separate models on $1{,}280$K instances for each scale and evaluate them on $1{,}280$ test instances. For TSP-500 and TSP-1000, we adopt zero-shot cross-scale evaluation, an established protocol in neural CO for assessing whether a solver trained on moderate-size instances remains effective as the problem scale increases (Fu et al., 2021; Drakulic et al., 2023; Luo et al., 2023; Wang et al., 2025a). In particular, prior work has directly evaluated models trained on TSP-100 on substantially larger TSP instances, including TSP-500 and TSP-1000 (Wang et al., 2025a). This setting is especially relevant to supervised neural solvers, since scale-specific training requires constructing a new labeled dataset at each target scale, while obtaining the large number of pre-computed optimal or high-quality solutions required for supervision becomes increasingly costly on large TSP instances (Fu et al., 2021). Following this established setting, we train DriftCO on TSP-100 and directly evaluate it on 128 TSP-500 and 128 TSP-1000 test instances, without target-scale supervision or fine-tuning. Reference solutions for the synthetic datasets are obtained using Concorde (Applegate et al., 2006). We further evaluate the TSP-100 model on TSPLIB benchmark instances, with detailed results reported in Appendix F.2.

Evaluation metrics. Following prior work (Sun & Yang, 2023; Li et al., 2024; Wang et al., 2025b), we evaluate performance using three metrics: 1) *Tour length*, the average length of the predicted tours across test instances; 2) *reference gap*, the average relative gap with respect to the reference solutions; and 3) *running time*, the total wall-clock time required to solve all test instances.

Setting. For the evaluated DM-based baselines, we follow DIFUSCO (Sun & Yang, 2023) and set the test-time batch size to 1. Detailed descriptions of the baselines and their inference configurations are provided in Appendix F.1. Following common practice in neural TSP solvers (Sun & Yang, 2023; Li et al., 2024; Ma et al., 2025a), all methods use the same decoding procedure described in Appendix E, followed by 2-Opt local search (Lin & Kernighan, 1973). For TSP-500 and TSP-1000, we follow DIFUSCO (Sun & Yang, 2023) and construct sparse graphs using the $k = 50$ nearest neighbors of each node. Since DEITSP (Wang et al., 2025a) does not support sparse graphs, we use its dense-graph implementation.

Main results. Table 1 reports the running time on TSP-50 and TSP-100. Since the evaluated DM-based baselines can generally attain the reference solution quality at these scales, we compare the time required to reach that quality. For each method, we report the inference configuration that achieves the reference quality with the lowest observed running time. DriftCO is the fastest among all compared methods, requiring only 13s on TSP-50 and 38s on TSP-100.

For TSP-500 and TSP-1000, all evaluated DM-based solvers directly transfer their TSP-100 models without target-scale supervision or fine-tuning, providing a unified zero-shot cross-scale evaluation. Table 2 reports the corresponding results. DriftCO achieves the best solution quality among the compared learning-based methods, with gaps of 2.23% and 3.24% on TSP-500 and TSP-1000, respectively, while requiring only 36s and 157s. These results demonstrate strong zero-shot cross-scale generalization while preserving the efficiency of one-step generation.

5.2 EXPERIMENTS FOR MIS

Datasets. Following prior work (Li et al., 2018; 2024; 2025b; Wang et al., 2025b; Ma et al., 2025a), we evaluate DriftCO on two datasets for MIS: RB graphs (Zhang et al., 2023) and Erdős–Rényi (ER) graphs (Erdős & Rényi, 1960). For RB graphs, we randomly generate instances with the number of vertices uniformly sampled from 200 to 300. The training set contains 90,000 instances, and the test set contains 500 instances. For the in-distribution evaluation, all DM-based solvers use

Table 1: Running time in seconds to attain the reference solution quality on TSP-50 and TSP-100.

Instance	DIFUSCO	Fast T2T	DEITSP	StruDiCO	COExpander	**DriftCO**
TSP-50	189	58	64	23	30	**13**
TSP-100	654	64	366	42	77	**38**

Table 2: Zero-shot cross-scale results on TSP-500 and TSP-1000. $*$ marks the reference for gap computation. Non-DM results are from (Wang et al., 2025a).

Method	Type	TSP-500			TSP-1000		
		Length↓	Gap (%)↓	Time↓	Length↓	Gap (%)↓	Time↓
Concorde*	Exact	16.55	0.00	35.12m	23.12	0.00	6.35h
2-Opt	Heuristic	17.99	8.70	20s	25.23	9.13	64s
AM	Non-DM	19.53	18.03	21.99m	29.90	29.24	1.64h
POMO	Non-DM	19.52	17.98	1.33m	30.51	31.96	8.80m
Sym-NCO	Non-DM	19.84	19.93	1.35m	31.47	36.13	8.88m
Pointerformer	Non-DM	18.62	12.53	1.35m	27.89	20.66	6.89m
ELG	Non-DM	17.73	7.16	1.94m	25.74	11.36	3.99m
HierTSP	Non-DM	18.54	12.08	1.50m	28.39	22.79	9.43m
BQ-NCO	Non-DM	17.21	4.02	7.05m	24.09	4.19	23.43m
DIFUSCO	DM-based	16.95	2.41	226s	23.95	3.58	626s
Fast T2T	DM-based	17.07	3.14	86s	24.13	4.36	605s
DEITSP	DM-based	16.95	2.41	117s	24.04	3.97	672s
StruDiCO	DM-based	17.49	5.67	79s	24.76	7.09	355s
COExpander	DM-based	17.27	4.35	62s	24.68	6.74	181s
DriftCO (Ours)	**Drifting**	**16.92**	**2.23**	**36s**	**23.87**	**3.24**	**157s**

their corresponding RB-trained models on the RB test set. To complement the cross-scale evaluation on TSP, we further assess zero-shot transfer under a simultaneous distribution and scale shift. Specifically, we directly evaluate the same RB-trained models on the ER instances from (Qiu et al., 2022), which contain 700–800 nodes with edge probability 0.15. No ER-specific supervision or fine-tuning is used for any evaluated DM-based solver. Reference solutions for both RB and ER graphs are obtained using KaMIS (Lamm et al., 2016).

Evaluation metrics. Following prior work (Sun & Yang, 2023; Li et al., 2024; Wang et al., 2025b), we report three metrics: 1) *Size*, the average size of the predicted maximal independent sets; 2) *reference gap*, the average relative gap with respect to the reference solutions; and 3) *running time*, the total wall-clock time required to solve all test instances. Gap and running time are computed as in the TSP experiments.

Setting. DEITSP (Wang et al., 2025a) is excluded from the MIS comparison because its released implementation does not support MIS. Following prior MIS solvers that combine learned predictions with local improvement (Li et al., 2018; Andrade et al., 2012), we apply the same ARW-style local search procedure to all evaluated DM-based methods as post-processing. The procedure combines greedy augmentation, $(1, 2)$-swap operations on 1-tight vertices, and random perturbations. This shared post-processing provides a consistent comparison of the candidate generation mechanisms.

Main results. Table 3 reports the results on RB and ER graphs. On the in-distribution RB-[200–300] test set, DriftCO achieves the best solution quality among the compared learning-based methods, with a gap of 2.63%, while requiring only 20s. On ER-[700–800], all evaluated DM-based solvers are transferred directly from RB without fine-tuning. DriftCO achieves the best solution quality among the compared learning-based methods, with a gap of 10.87%, while retaining efficient inference at 28s. These results show strong zero-shot generalization beyond both the graph distribution and problem sizes seen during training.

5.3 Candidate Diversity under Different Inference Mechanisms

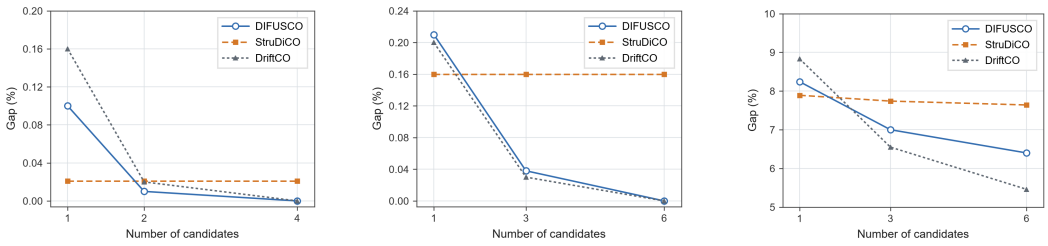
One motivation of DriftCO is to retain useful candidate diversity without relying on multi-step stochastic trajectories at inference time. We therefore study how different generation mechanisms

Table 3: MIS results on RB-[200–300] and zero-shot transfer to ER-[700–800]. $*$ marks the reference for gap computation. Non-DM neural baseline results are reported from (Li et al., 2024).

Method	Type	RB-[200–300]			ER-[700–800]		
		Size↑	Gap (%)↓	Time↓	Size↑	Gap (%)↓	Time↓
KaMIS*	Heuristic	20.14	0.00	1.42h	44.87	0.00	53.01m
ARW Local Search	Heuristic	17.84	11.42	10s	35.42	21.06	6s
Intel-G	Non-DM	–	–	–	34.86	22.31	6.06m
DIMES	Non-DM	–	–	–	38.24	14.78	6.12m
Intel-TS	Non-DM	18.47	8.29	13.07m	38.80	13.53	20.00m
DGL	Non-DM	17.36	13.80	12.78m	37.26	16.96	22.71m
DIFUSCO	DM-based	19.45	3.42	192s	39.82	11.25	496s
Fast T2T	DM-based	19.58	2.78	46s	39.66	11.61	43s
StruDiCO	DM-based	19.56	2.87	22s	39.96	10.94	30s
COExpander	DM-based	19.26	4.36	89s	38.50	14.19	55s
DriftCO (Ours)	**Drifting**	**19.61**	**2.63**	**20s**	**39.99**	**10.87**	**28s**

benefit from increasing the number of candidates. Rather than measuring diversity solely through geometric distances between solutions, we use the improvement in final solution quality as the candidate budget increases as an operational measure of useful diversity. If additional candidates explore complementary high-quality neighborhoods, selecting the best candidate after decoding and local improvement should yield progressively better solutions.

We compare three solvers with distinct inference mechanisms. DIFUSCO uses 20-step stochastic denoising, allowing candidate variation to accumulate along independent multi-step trajectories. For StruDiCO, we evaluate its one-step setting with inference-stage boosting disabled, isolating the behavior of the accelerated one-step sampler. DriftCO generates every candidate with a single forward pass and promotes variation through same-instance attraction–repulsion during training.



(a) TSP-50. (b) TSP-100. (c) RB-[200–300].

Figure 1: Effect of increasing the number of candidates under different inference mechanisms.

As shown in Figure 1, DIFUSCO consistently benefits from increasing the candidate budget, consistent with multi-step stochastic denoising providing useful variation across candidates. In the one-step StruDiCO setting, the gains from additional candidates are notably smaller, suggesting that compressing the inference trajectory can reduce the variation available to best-of-many selection. Despite using no denoising trajectory, DriftCO continues to benefit from additional candidates across all three settings. These results support the role of training-time attraction–repulsion in preserving useful candidate variation under one-step generation.

We further examine the contribution of attraction–repulsion learning and the structural cosine kernel in Appendix F.3.

6 Conclusion

We presented DriftCO, a drifting model-based CO solver that enables one-step candidate generation through training-time attraction–repulsion. Experiments on TSP and MIS demonstrate effective solution generation, efficient one-step inference, and zero-shot generalization across problem scales and graph distributions, supporting drifting model-based training as an efficient approach to generative CO solving.

AI USE STATEMENT

Generative AI tools, including large language models (LLMs), were used during this work to assist with surveying and summarizing related literature, and drafting and editing parts of the manuscript for clarity and readability. All AI-assisted literature-related information and manuscript edits were independently reviewed and verified by the authors. Experimental results reported in the paper were obtained from our own implementations and experiments, and claims about prior work were checked against the corresponding original sources. The authors made all final decisions regarding the content and presentation of the paper and take full responsibility for this work.

ETHICS STATEMENT

This work studies algorithmic methods for combinatorial optimization using synthetic and publicly available benchmark datasets. It does not involve human participants, personally identifiable information, or sensitive data. We are not aware of any specific ethical concerns arising from this work beyond standard considerations associated with computational research.

REPRODUCIBILITY STATEMENT

We provide implementation details, experimental settings, and evaluation protocols in the main text and appendix to support reproduction of the reported results.

REFERENCES

Sungsoo Ahn, Younggyo Seo, and Jinwoo Shin. Learning what to defer for maximum independent sets. In *ICML*, 2020.

Diogo V Andrade, Mauricio GC Resende, and Renato F Werneck. Fast local search for the maximum independent set problem. *Journal of Heuristics*, 18(4):525–547, 2012.

David Applegate, Robert Bixby, Vasek Chvatal, and William Cook. Concorde TSP solver, 2006. `https://www.math.uwaterloo.ca/tsp/concorde/index.html`.

Mickael Basson and Philippe Preux. IDEQ: An improved diffusion model for the TSP. *arXiv preprint arXiv:2412.13858*, 2024.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *ICLR*, 2017.

Federico Berto, Chuanbo Hua, Laurin Luttmann, Jiwoo Son, Junyoung Park, Kyuree Ahn, Changhyun Kwon, Lin Xie, and Jinkyoo Park. PARCO: Parallel autoregressive models for multi-agent combinatorial optimization. In *NeurIPS*, 2025a.

Federico Berto, Chuanbo Hua, Nayeli Gast Zepeda, André Hottung, Niels Wouda, Leon Lan, Junyoung Park, Kevin Tierney, and Jinkyoo Park. RouteFinder: Towards foundation models for vehicle routing problems. *Transactions on Machine Learning Research*, 2025b.

Federico Berto et al. RL4CO: An extensive reinforcement learning for combinatorial optimization benchmark. In *SIGKDD*, 2025c.

Jieyi Bi, Yining Ma, Jianan Zhou, Wen Song, Zhiguang Cao, Yaoxin Wu, and Jie Zhang. Learning to handle complex constraints for vehicle routing problems. In *NeurIPS*, 2024.

Jieyi Bi, Zhiguang Cao, Jianan Zhou, Wen Song, Yaoxin Wu, Jie Zhang, Yining Ma, and Cathy Wu. Towards efficient constraint handling in neural solvers for routing problems. In *ICLR*, 2026.

Fanchen Bu, Hyeonsoo Jo, Soo Yong Lee, Sungsoo Ahn, and Kijung Shin. Tackling prevalent conditions in unsupervised combinatorial optimization: Cardinality, minimum, covering, and more. In *ICML*, 2024.

Felix Chalumeau, Refiloe Shabe, Noah De Nicola, Arnu Pretorius, Thomas D Barrett, and Nathan Grinsztajn. Memory-enhanced neural solvers for routing problems. In *NeurIPS*, 2025.

Lvda Chen, Yang Li, and Junchi Yan. MaskCO: Masked generation drives effective representation learning and exploiting for combinatorial optimization. In *ICLR*, 2026.

Ruogu Chen and Jie Han. EDISCO: Equivariant continuous-time categorical diffusion for geometric combinatorial optimization, 2026. OpenReview. Available at: `https://openreview.net/forum?id=A1JfGuXJvM`.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *NeurIPS*, 2019.

Yuanyao Chen, Rongsheng Chen, Fu Luo, and Zhenkun Wang. Improving generalization of neural combinatorial optimization for vehicle routing problems via test-time projection learning. In *NeurIPS*, 2025.

Gobinda G Chowdhury. *Introduction to modern information retrieval*. Facet publishing, 2010.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. BQ-NCO: Bisimulation quotienting for efficient neural combinatorial optimization. In *NeurIPS*, 2023.

Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. GOAL: A generalist combinatorial optimization agent learner. In *ICLR*, 2025.

Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5(1):17–61, 1960.

Mingfeng Fan, Jianan Zhou, Yifeng Zhang, Yaoxin Wu, Jinbiao Chen, and Guillaume Adrien Sartoretti. Preference-driven multi-objective combinatorial optimization with conditional computation. In *NeurIPS*, 2025.

Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. INViT: A generalizable routing problem solver with invariant nested view transformer. In *ICML*, 2024.

Shengyu Feng, Weiwei Sun, Shanda Li, Ameet Talwalkar, and Yiming Yang. FrontierCO: Real-world and large-scale evaluation of machine learning solvers for combinatorial optimization. In *ICLR*, 2026.

Benjy Friedmann and Nadav Dym. Transport, don't generate: Deterministic geometric flows for combinatorial optimization. *arXiv preprint arXiv:2602.10794*, 2026.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large TSP instances. In *AAAI*, 2021.

Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. *arXiv preprint arXiv:2308.14104*, 2023.

Songqun Gao, Zanxi Ruan, Patrick Floor, Marco Roveri, Luigi Palopoli, and Daniele Fontanelli. Learning to solve orienteering problem with time windows and variable profits. In *ICLR*, 2026.

Andoni Irazusta Garmendia. A first guess is rarely the final answer: Learning to search in the travelling salesperson problem. *arXiv preprint arXiv:2604.06940*, 2026.

Andoni Irazusta Garmendia, Josu Ceberio, and Alexander Mendiburu. Enabling population-based architectures for neural combinatorial optimization. *arXiv preprint arXiv:2601.08696*, 2026.

Yong Liang Goh, Zhiguang Cao, Yining Ma, Yanfei Dong, Mohammed Haroon Dupty, and Wee Sun Lee. Hierarchical neural constructive solver for real-world tsp scenarios. In *SIGKDD*, 2024.

Yong Liang Goh, Zhiguang Cao, Yining Ma, Jianan Zhou, Mohammed Haroon Dupty, and Wee Sun Lee. SHIELD: Multi-task multi-distribution vehicle routing solver with sparsity and hierarchy. In *ICML*, 2025.

Shengda Gu, Kai Li, Junliang Xing, Yifan Zhang, and Jian Cheng. Synergizing reinforcement learning and genetic algorithms for neural combinatorial optimization. *arXiv preprint arXiv:2506.09404*, 2025.

Shuangchun Gui, Suyu Liu, Xuehe Wang, and Zhiguang Cao. Chain-of-context learning: Dynamic constraint understanding for multi-task VRPs. In *ICLR*, 2026.

Ziao Guo, Yang Li, Shiyue Wang, and Junchi Yan. ConRep4CO: Contrastive representation learning of combinatorial optimization instances across types. In *ICLR*, 2026.

Seong-Hyun Hong, Hyun-Sung Kim, Zian Jang, Deunsol Yoon, Hyungseok Song, and Byung-Jun Lee. Unsupervised training of diffusion models for feasible solution generation in neural combinatorial optimization. *arXiv preprint arXiv:2411.00003*, 2024.

Heiko Hoppe, Léo Baty, Louis Bouvier, Axel Parmentier, and Maximilian Schiffer. Structured reinforcement learning for combinatorial decision-making. In *NeurIPS*, 2025.

André Hottung, Mridul Mahajan, and Kevin Tierney. PolyNet: Learning diverse solution strategies for neural combinatorial optimization. In *ICLR*, 2025a.

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

Junwei Huang, Zhiqing Sun, and Yiming Yang. Accelerating diffusion-based combinatorial optimization solvers by progressive distillation. *arXiv preprint arXiv:2308.06644*, 2023.

Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. Large language models as end-to-end combinatorial optimization solvers. In *NeurIPS*, 2025.

Yan Jin, Yuandong Ding, Xuanhao Pan, Kun He, Li Zhao, Tao Qin, Lei Song, and Jiang Bian. Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem. In *AAAI*, 2023.

Yaochu Jin, Xueming Yan, Shiqing Liu, and Xiangyu Wang. A unified framework for combinatorial optimization based on graph neural networks. *arXiv preprint arXiv:2406.13125*, 2024.

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.

Nikolaos Karalias, Akbar Rafiey, Yifei Xu, Zhishang Luo, Behrooz Tahmasebi, Connie Jiang, and Stefanie Jegelka. Geometric algorithms for neural combinatorial optimization with constraints. In *NeurIPS*, 2025.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NeurIPS*, 2017.

Hyeonah Kim, Minsu Kim, Sungsoo Ahn, and Jinkyoo Park. Symmetric replay training: Enhancing sample efficiency in deep reinforcement learning for combinatorial optimization. In *ICML*, 2024.

Hyeonah Kim, Sanghyeok Choi, Jiwoo Son, Jinkyoo Park, and Changhyun Kwon. Neural genetic search in discrete spaces. In *ICML*, 2025.

Minsu Kim, Sanghyeok Choi, Hyeonah Kim, Jiwoo Son, Jinkyoo Park, and Yoshua Bengio. Ant colony sampling with GFlowNets for combinatorial optimization. In *AISTATS*, 2022a.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. In *NeurIPS*, 2022b.

Detian Kong, Yining Ma, Zhiguang Cao, Tianshu Yu, and Jianhua Xiao. Efficient neural collaborative search for pickup and delivery problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):11019–11034, 2024.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *ICLR*, 2019.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *NeurIPS*, 2020.

Chieh-Hsin Lai, Bac Nguyen, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yuki Mitsufuji, Stefano Ermon, and Molei Tao. A unified view of drifting and score-based models. *arXiv preprint arXiv:2603.07514*, 2026.

Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F Werneck. Finding near-optimal independent sets at scale. In *ALENEX*, 2016.

Haoyu Lei, Kaiwen Zhou, Yinchuan Li, Zhitang Chen, and Farzan Farnia. Boosting cross-problem generalization in diffusion-based neural combinatorial solver via inference time adaptation. In *AAAI*, 2026.

Ke Li, Fei Liu, Zhenkun Wang, and Qingfu Zhang. Destroy and repair using hyper-graphs for routing. In *AAAI*, 2025a.

Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. T2T: From distribution learning in training to gradient search in testing for combinatorial optimization. In *NeurIPS*, 2023.

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 *NeurIPS*, 2024.

Yang Li, Lvda Chen, Haonan Wang, Runzhong Wang, and Junchi Yan. Generation as search operator for test-time scaling of diffusion-based combinatorial optimization. In *NeurIPS*, 2025b.

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 *ICLR*, 2025c.

Yuanshu Li, Di Wang, Wei Du, Xuan Wu, Peng Zhao, Yubin Xiao, and You Zhou. Efficient few-step solution generation via discrete flow matching for combinatorial optimization. In *AAAI*, 2026.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *NeurIPS*, 2018.

Zijun Liao, Jinbiao Chen, Debing Wang, Zizhen Zhang, and Jiahai Wang. BOPO: Neural combinatorial optimization via best-anchored and objective-guided preference optimization. In *ICML*, 2025.

Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

Suyu Liu, Zhiguang Cao, Nan Yin, and Yew-Soon Ong. Scale-Net: A hierarchical U-Net framework for cross-scale generalization in multi-task vehicle routing. In *AAAI*, 2026.

Hao Lu, Xingwen Zhang, and Shuang Yang. A learning-based iterative method for solving vehicle routing problems. In *ICLR*, 2019.

Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *NeurIPS*, 2023.

Fu Luo, Xi Lin, Yaoxin Wu, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Qingfu Zhang. Boosting neural combinatorial optimization for large-scale vehicle routing problems. In *ICLR*, 2025a.

Fu Luo, Xi Lin, Mengyuan Zhong, Fei Liu, Zhenkun Wang, Jianyong Sun, and Qingfu Zhang. Learning to insert for constructive neural vehicle routing solver. In *NeurIPS*, 2025b.

Fu Luo, Yaoxin Wu, Zhi Zheng, and Zhenkun Wang. Rethinking neural combinatorial optimization for vehicle routing problems with different constraint tightness degrees. In *NeurIPS*, 2025c.

Qing Luo, Fu Luo, Ke Li, and Zhenkun Wang. Efficient decoder scaling strategy for neural routing solvers. *arXiv preprint arXiv:2603.00430*, 2026.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. COExpander: Adaptive solution expansion for combinatorial optimization. In *ICML*, 2025a.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. ML4CO-Bench-101: Benchmark machine learning for classic combinatorial problems on graphs. In *NeurIPS*, 2025b.

Yining Ma, Zhiguang Cao, and Yeow Meng Chee. Learning to search feasible and infeasible regions of routing problems with flexible neural k-Opt. In *NeurIPS*, 2023.

Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. In *NeurIPS*, 2023.

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

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 *ICLR*, 2025.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. DIMES: A differentiable meta solver for combinatorial optimization problems. In *NeurIPS*, 2022.

Robbert Reijnen, Yaoxin Wu, Zaharah Bukhsh, and Yingqian Zhang. Graph-supported dynamic algorithm configuration for multi-objective combinatorial optimization. In *ICML*, 2025.

Filip Rydin, Attila Lischka, Jiaming Wu, Morteza Haghir Chehreghani, and Balázs Kulcsár. Beyond simple graphs: Neural multi-objective routing on multigraphs. In *ICLR*, 2026.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *ICML*, 2024.

Sebastian Sanokowski, Wilhelm Franz Berghammer, Haoyu Peter Wang, Martin Ennemoser, Sepp Hochreiter, and Sebastian Lehner. Scalable discrete diffusion samplers: Combinatorial optimization and statistical physics. In *ICLR*, 2025.

Jun Pyo Seo. Blackout DIFUSCO. *arXiv preprint arXiv:2502.05221*, 2025.

Joan Salvà Soler and Günther R Raidl. A denoising diffusion-based evolutionary algorithm framework: Application to the maximum independent set problem. *arXiv preprint arXiv:2510.08627*, 2025.

Jiwoo Son, Minsu Kim, Sanghyeok Choi, Hyeonah Kim, and Jinkyoo Park. Equity-transformer: Solving NP-hard min-max routing problems as sequential generation with equity context. In *AAAI*, 2024.

Hyungseok Song, Deunsol Yoon, Kanghoon Lee, Han-Seul Jeong, Soonyoung Lee, and Woohyung Lim. CADO: From imitation to cost minimization for heatmap-based solvers in combinatorial optimization. *arXiv preprint arXiv:2602.08210*, 2026.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *NeurIPS*, 2023.

Sobihan Surendran, Adeline Fermanian, and Sylvain Le Corff. Diffusion-driven latent guided sampling for neural combinatorial optimization. In *ICLR 2026 Workshop on ReALM-GEN*, 2026.

Federico Julian Camerota Verdù, Lorenzo Castelli, and Luca Bortolussi. Scaling combinatorial optimization neural improvement heuristics with online search and adaptation. In *AAAI*, 2025.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NeurIPS*, 2015.

Chenguang Wang, Zhouliang Yu, Stephen McAleer, Tianshu Yu, and Yaodong Yang. ASP: Learn a universal neural solver! *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6): 4102–4114, 2024.

Mingzhao Wang, You Zhou, Zhiguang Cao, Yubin Xiao, Xuan Wu, Wei Pang, Yuan Jiang, Hui Yang, Peng Zhao, and Yuanshu Li. An efficient diffusion-based non-autoregressive solver for traveling salesman problem. In *SIGKDD*, 2025a.

Wentao Wang, Lifeng Han, and Guangyu Zou. Cluster-aware attention-based deep reinforcement learning for pickup and delivery problems. *arXiv preprint arXiv:2603.10053*, 2026a.

Yu Wang, Yang Li, Junchi Yan, and Yi Chang. StruDiCO: Structured denoising diffusion with gradient-free inference-stage boosting for memory and time efficient combinatorial optimization. In *NeurIPS*, 2025b.

Yu Wang, Yang Li, Jiale Ma, Junchi Yan, and Yi Chang. Native adaptive solution expansion for diffusion-based combinatorial optimization. In *ICLR*, 2026b.

Zhenwei Wang, Tiehua Zhang, Ning Xue, Ender Ozcan, Ling Wang, and Ruibin Bai. Constraints matrix diffusion based generative neural solver for vehicle routing problems. *arXiv preprint arXiv:2603.07568*, 2026c.

Lowell Weissman, Michael Krumdick, and A Lynn Abbott. Complexity scaling laws for neural models using combinatorial optimization. In *NeurIPS*, 2025.

Yifan Xia, Xianliang Yang, Zichuan Liu, Zhihao Liu, Lei Song, and Jiang Bian. Position: Rethinking post-hoc search-based neural approaches for solving large-scale traveling salesman problems. In *ICML*, 2024.

Yubin Xiao, Di Wang, Boyang Li, Huanhuan Chen, Wei Pang, Xuan Wu, Hao Li, Dong Xu, Yanchun Liang, and You Zhou. Reinforcement learning-based nonautoregressive solver for traveling salesman problems. *IEEE Transactions on Neural Networks and Learning Systems*, 36(7):13402–13416, 2024a.

Yubin Xiao, Di Wang, Boyang Li, Mingzhao Wang, Xuan Wu, Changliang Zhou, and You Zhou. Distilling autoregressive models to obtain high-performance non-autoregressive solvers for vehicle routing problems with faster inference speed. In *AAAI*, 2024b.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. In *NeurIPS*, 2021.

Yudong W Xu, Wenhao Li, Scott Sanner, and Elias B Khalil. Large neighborhood search meets iterative neural constraint heuristics. *arXiv preprint arXiv:2603.20801*, 2026.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. DeepACO: Neural-enhanced ant systems for combinatorial optimization. In *NeurIPS*, 2023.

Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. GLOP: Learning global partition and local construction for solving large-scale routing problems in real-time. In *AAAI*, 2024.

Boyu Zhang, Aocheng Shen, Bing Liu, Qiankun Zhang, Bin Yuan, Jing Wang, Shenghao Liu, and Xianjun Deng. DiMa: Understanding the hardness of online matching problems via diffusion models. In *ICML*, 2025a.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with GFlowNets. In *NeurIPS*, 2023.

Ni Zhang and Zhiguang Cao. Hybrid-balance GFlowNet for solving vehicle routing problems. In *NeurIPS*, 2025.

Ni Zhang, Jingfeng Yang, Zhiguang Cao, and Xu Chi. Adversarial generative flow network for solving vehicle routing problems. In *ICLR*, 2025b.

Hang Zhao, Kexiong Yu, Yuhang Huang, Renjiao Yi, Chenyang Zhu, and Kai Xu. DISCO: Efficient diffusion solver for large-scale combinatorial optimization problems. *Graphical Models*, 141: 101284, 2025.

Zhi Zheng, Changliang Zhou, Xialiang Tong, Mingxuan Yuan, and Zhenkun Wang. UDC: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems. In *NeurIPS*, 2024.

Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Xu Chi. MVMoE: Multi-task vehicle routing solver with mixture-of-experts. In *ICML*, 2024a.

Jianan Zhou, Yaoxin Wu, Zhiguang Cao, Wen Song, Jie Zhang, and Zhiqi Shen. Collaboration! Towards robust neural methods for routing problems. In *NeurIPS*, 2024b.

Shipei 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 *AAAI*, 2025.

A Network Architecture Details

Following prior work (Sun & Yang, 2023), we instantiate the generator f_θ with an anisotropic graph neural network (AGNN) architecture adapted to our setting.

A.1 Input Embedding Layer

Given the node vector $\mathbf{h} \in \mathbb{R}^{N\times 2}$ and the edge vector $\mathbf{e} \in \mathbb{R}^{N\times N}$, where N denotes the number of nodes in the graph, we compute sinusoidal features for each input element. Since DriftCO directly generates solution heatmaps without iterative denoising, it does not require diffusion timestep embeddings.

For each coordinate $\mathbf{h}_{i,j}$, we use a $d/2$-dimensional sinusoidal embedding:

$$\tilde{\mathbf{h}}_{i,j} = \mathrm{concat}_{m=0}^{d/4-1}\left(\sin\frac{\mathbf{h}_{i,j}}{K^{\frac{2m}{d/2}}}, \cos\frac{\mathbf{h}_{i,j}}{K^{\frac{2m}{d/2}}}\right) \in \mathbb{R}^{d/2}, \tag{11}$$

$$\tilde{\mathbf{h}}_i = \mathrm{concat}\left(\tilde{\mathbf{h}}_{i,0}, \tilde{\mathbf{h}}_{i,1}\right) \in \mathbb{R}^{d}. \tag{12}$$

For each edge element $\mathbf{e}_{i,k}$, we use a d-dimensional sinusoidal embedding:

$$\tilde{\mathbf{e}}_{i,k} = \mathrm{concat}_{m=0}^{d/2-1}\left(\sin\frac{\mathbf{e}_{i,k}}{K^{\frac{2m}{d}}}, \cos\frac{\mathbf{e}_{i,k}}{K^{\frac{2m}{d}}}\right) \in \mathbb{R}^{d}, \tag{13}$$

$$\tilde{\mathbf{e}}_i = \mathrm{concat}\left(\tilde{\mathbf{e}}_{i,0}, \ldots, \tilde{\mathbf{e}}_{i,N-1}\right) \in \mathbb{R}^{d\times N}. \tag{14}$$

Here, d is the embedding dimension, $K = 10$, $j \in \{0,1\}$, $k \in \{0,\ldots,N-1\}$, and $\mathrm{concat}(\cdot)$ denotes concatenation.

The input features for the graph convolution layers are then computed as

$$\mathbf{h}_i^0 = \mathbf{W}_1^0\tilde{\mathbf{h}}_i \in \mathbb{R}^d, \tag{15}$$

$$\mathbf{e}_i^0 = \mathbf{W}_2^0\tilde{\mathbf{e}}_i \in \mathbb{R}^{d\times N}. \tag{16}$$

Here, superscripts indicate layer indices, and $\mathbf{W}_1^0, \mathbf{W}_2^0 \in \mathbb{R}^{d\times d}$ are learnable parameter matrices. For TSP, the node vector $\mathbf{h}$ is initialized with the node coordinates $\mathbf{c}$, while the edge vector $\mathbf{e}$ is initialized with per-edge noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For MIS, the node vector $\mathbf{h}$ is initialized with per-node noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, while the edge vector $\mathbf{e}$ is initialized to zero. Independent noise samples provide the stochastic inputs used to generate multiple candidates for the same problem instance.

A.2 Graph Convolution Layer

The cross-layer convolution operation is formulated as

$$\mathbf{h}_i^{l+1} = \mathbf{h}_i^l + \mathrm{ReLU}\left(\mathrm{BN}\left(\mathbf{W}_1^l\mathbf{h}_i^l + \sum_{j\sim i}\boldsymbol{\eta}_{i,j}^l \odot \mathbf{W}_2^l\mathbf{h}_j^l\right)\right) \in \mathbb{R}^d, \tag{17}$$

$$\mathbf{e}_{i,j}^{l+1} = \mathbf{e}_{i,j}^l + \mathrm{ReLU}\left(\mathrm{BN}\left(\mathbf{W}_3^l\mathbf{e}_{i,j}^l + \mathbf{W}_4^l\mathbf{h}_i^l + \mathbf{W}_5^l\mathbf{h}_j^l\right)\right) \in \mathbb{R}^d, \tag{18}$$

$$\boldsymbol{\eta}_{i,j}^l = \frac{\sigma(\mathbf{e}_{i,j}^l)}{\sum_{j'\sim i}\sigma(\mathbf{e}_{i,j'}^l)} \in \mathbb{R}^d. \tag{19}$$

Here, $\mathbf{h}_i^l$ and $\mathbf{e}_{i,j}^l$ denote the node and edge feature vectors at layer l, respectively. $\mathbf{W}_1^l, \ldots, \mathbf{W}_5^l \in \mathbb{R}^{d\times d}$ are learnable parameter matrices, $\boldsymbol{\eta}_{i,j}^l$ denotes the attention weight, BN denotes batch normalization, and σ denotes the sigmoid function.

A.3 Output Layer

The output heatmap is obtained from the final-layer edge features for TSP and node features for MIS:

$$\hat{x}_{i,j} = \mathbf{W}_e \,\mathrm{ReLU}\left(\mathrm{BN}(\mathbf{e}_{i,j}^{L})\right), \tag{20}$$

$$\hat{x}_i = \mathbf{W}_n \,\mathrm{ReLU}\left(\mathrm{BN}(\mathbf{h}_i^{L})\right), \tag{21}$$

where L denotes the number of graph convolution layers and $\mathbf{W}_e, \mathbf{W}_n \in \mathbb{R}^{1\times d}$ are learnable output projections.

A.4 Hyperparameters

For both TSP and MIS, we use a 12-layer graph neural network with embedding dimension $d = 256$.

B Training Procedure

Algorithm 1 provides the complete training procedure of DriftCO. For each graph instance, we generate K candidate heatmaps from independent noise samples, compute the instance-wise drifting field using the single reference solution and the remaining generated candidates, and regress each heatmap toward its stop-gradient drifted target. The computation of the drifting field is detailed in Appendix C.

Algorithm 1 Training of DriftCO

Require: Generator f_θ, problem distribution $\mathcal{D}$, batch size B, samples per instance K, temperature τ, learning rate η, total iterations S

for $s = 1, \dots, S$ **do**
  Sample a mini-batch $\{(G_b, \mathbf{y}_b^*)\}_{b=1}^{B} \sim \mathcal{D}$
  for $b = 1, \dots, B$ **do**
    for $k = 1, \dots, K$ **do**
      Sample noise $\mathbf{z}_b^{(k)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
      $\hat{\mathbf{x}}_b^{(k)} \leftarrow f_\theta(\mathbf{z}_b^{(k)}, G_b)$
    end for
    for $k = 1, \dots, K$ **do**
      $\widehat{\mathbf{V}}_b^{(k)} \leftarrow \texttt{compute-V}\Big(\hat{\mathbf{x}}_b^{(k)}, \mathbf{y}_b^*, \{\hat{\mathbf{x}}_b^{(j)}\}_{j\neq k}, \tau\Big)$
      $\mathbf{t}_b^{(k)} \leftarrow \mathrm{sg}\Big(\hat{\mathbf{x}}_b^{(k)} + \widehat{\mathbf{V}}_b^{(k)}\Big)$
    end for
  end for
  $\mathcal{L}(\theta) \leftarrow \frac{1}{B}\sum_{b=1}^{B} \frac{1}{K}\sum_{k=1}^{K} \big\|\hat{\mathbf{x}}_b^{(k)} - \mathbf{t}_b^{(k)}\big\|_2^2$
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$
end for
return θ

C Pseudocode for Computing the Drifting Field V

Algorithm 2 provides the pseudocode for computing the empirical instance-wise drifting field used during training. It implements the finite-sample estimator of the attraction and repulsion terms defined in Eqs. 3–5. For a generated heatmap $\hat{\mathbf{x}}_b^{(k)}$, the attraction component is determined directly by the single reference solution $\mathbf{y}_b^*$, while the repulsion component is estimated from the remaining same-instance generated heatmaps $\{\hat{\mathbf{x}}_b^{(j)}\}_{j\neq k}$. The structural cosine kernel in Eq. 7 assigns larger repulsion weights to generated candidates occupying similar structural neighborhoods. The final drifting field is obtained by subtracting the estimated repulsion from the single-reference attraction.

Algorithm 2 `compute-V`: Instance-wise Conditional Drifting Field

Require: Heatmap $\hat{\mathbf{x}}_b^{(k)} \in \mathbb{R}^D$, reference solution $\mathbf{y}_b^*$, same-instance generated heatmaps $\{\hat{\mathbf{x}}_b^{(j)}\}_{j\neq k}$, temperature τ

$\mathbf{V}^+ \leftarrow \mathbf{y}_b^* - \hat{\mathbf{x}}_b^{(k)}$
for each $j \neq k$ **do**
  $w_j^- \leftarrow \kappa_\tau\left(\hat{\mathbf{x}}_b^{(k)}, \hat{\mathbf{x}}_b^{(j)}\right)$
end for
$\mathbf{V}^- \leftarrow \dfrac{1}{\sum_{j\neq k} w_j^-} \sum_{j\neq k} w_j^- \left(\hat{\mathbf{x}}_b^{(j)} - \hat{\mathbf{x}}_b^{(k)}\right)$
return $\mathbf{V} \leftarrow \mathbf{V}^+ - \mathbf{V}^-$

D Derivation of the Instance-wise Drifting Field

This section derives the single-reference instance-wise drifting field in Eq. 5. We first show that the normalized attraction term of the original drifting formulation reduces exactly to the displacement toward the single reference solution, and then derive the resulting combined attraction–repulsion field.

For an instance G, let $\mathbf{y}_G^*$ denote its single reference solution. The corresponding positive distribution can be represented as a point mass,

$$p^+(\mathbf{y} \mid G) = \delta(\mathbf{y} - \mathbf{y}_G^*). \tag{22}$$

Following the normalized attraction formulation of drifting models, we have

$$\mathbf{V}^+(\hat{\mathbf{x}}; G) = \frac{\mathbb{E}_{\mathbf{y}^+\sim p^+(\cdot|G)}\left[\kappa(\hat{\mathbf{x}}, \mathbf{y}^+)(\mathbf{y}^+ - \hat{\mathbf{x}})\right]}{\mathbb{E}_{\mathbf{y}^+\sim p^+(\cdot|G)}\left[\kappa(\hat{\mathbf{x}}, \mathbf{y}^+)\right]}. \tag{23}$$

Since $p^+(\cdot \mid G)$ contains only $\mathbf{y}_G^*$, Eq. 23 becomes

$$\mathbf{V}^+(\hat{\mathbf{x}}; G) = \frac{\kappa(\hat{\mathbf{x}}, \mathbf{y}_G^*)(\mathbf{y}_G^* - \hat{\mathbf{x}})}{\kappa(\hat{\mathbf{x}}, \mathbf{y}_G^*)} = \mathbf{y}_G^* - \hat{\mathbf{x}}, \tag{24}$$

which recovers the attraction field in Eq. 3. Thus, under single-reference supervision, the kernel normalization cancels exactly in the attraction term.

The repulsion field remains estimated over the conditional generated distribution $q_\theta(\cdot \mid G)$:

$$\mathbf{V}_{q_\theta}^-(\hat{\mathbf{x}}; G) = \frac{1}{Z_{q_\theta}(\hat{\mathbf{x}})} \mathbb{E}_{\tilde{\mathbf{x}}\sim q_\theta(\cdot|G)}\left[\kappa(\hat{\mathbf{x}}, \tilde{\mathbf{x}})(\tilde{\mathbf{x}} - \hat{\mathbf{x}})\right], \tag{25}$$

where

$$Z_{q_\theta}(\hat{\mathbf{x}}) = \mathbb{E}_{\tilde{\mathbf{x}}\sim q_\theta(\cdot|G)}\left[\kappa(\hat{\mathbf{x}}, \tilde{\mathbf{x}})\right]. \tag{26}$$

The total drifting field is defined by subtracting repulsion from attraction:

$$\mathbf{V}(\hat{\mathbf{x}}; G) = \mathbf{y}_G^* - \hat{\mathbf{x}} - \frac{1}{Z_{q_\theta}(\hat{\mathbf{x}})} \mathbb{E}_{\tilde{\mathbf{x}}\sim q_\theta(\cdot|G)}\left[\kappa(\hat{\mathbf{x}}, \tilde{\mathbf{x}})(\tilde{\mathbf{x}} - \hat{\mathbf{x}})\right]. \tag{27}$$

Using the definition of $Z_{q_\theta}(\hat{\mathbf{x}})$, the attraction term can equivalently be written as

$$\mathbf{y}_G^* - \hat{\mathbf{x}} = \frac{1}{Z_{q_\theta}(\hat{\mathbf{x}})} \mathbb{E}_{\tilde{\mathbf{x}}\sim q_\theta(\cdot|G)}\left[\kappa(\hat{\mathbf{x}}, \tilde{\mathbf{x}})(\mathbf{y}_G^* - \hat{\mathbf{x}})\right]. \tag{28}$$

Substituting Eq. 28 into Eq. 27 and using linearity of expectation gives

$$\mathbf{V}(\hat{\mathbf{x}}; G) = \frac{1}{Z_{q_\theta}(\hat{\mathbf{x}})} \mathbb{E}_{\tilde{\mathbf{x}}\sim q_\theta(\cdot|G)}\left[\kappa(\hat{\mathbf{x}}, \tilde{\mathbf{x}})(\mathbf{y}_G^* - \tilde{\mathbf{x}})\right], \tag{29}$$

which recovers Eq. 5.

E Decoding Method

TSP decoding. Let $\{\hat{x}_{ij}\}$ denote the edge confidence scores generated by DriftCO. Following prior work (Sun & Yang, 2023), we use greedy edge insertion for decoding. Specifically, all edges are ranked in decreasing order of $(\hat{x}_{ij} + \hat{x}_{ji})/\|\mathbf{c}_i - \mathbf{c}_j\|$, where $\mathbf{c}_i$ denotes the coordinates of node i. Edges are then inserted sequentially whenever they do not conflict with the current partial solution.

MIS decoding. Let $\{\hat{x}_i\}$ denote the node confidence scores generated by DriftCO. Following prior work (Sun & Yang, 2023), we use greedy decoding by ranking nodes in decreasing order of $\hat{x}_i$. A node is added to the partial independent set if none of its neighbors has already been selected.

F Supplementary Experiments

Training settings. Each training instance G_b is paired with exactly one high-quality reference solution $\mathbf{y}_b^*$, taken from the same datasets used by DIFUSCO (Sun & Yang, 2023) and Fast T2T (Li et al., 2024). No additional reference solutions are constructed for training. Training is conducted on 8 NVIDIA H100 GPUs, while all evaluations are performed on a single NVIDIA RTX 4090 GPU.

Training hyperparameters. Unless otherwise specified, we use $K = 8$ generated samples per instance to estimate the same-instance repulsion during training. For the structural cosine kernel, we use temperatures $\tau \in \{0.02, 0.05, 0.20\}$ and aggregate the drifting fields computed at the three temperatures. We use a cosine learning-rate schedule with an initial learning rate of 2×10^{-4}.

F.1 Baselines and Parameter Settings

We describe the generative CO solvers considered in this work and summarize the inference settings for the baselines included in our main empirical comparison. We additionally discuss several closely related recent methods that are not included in the main comparison because public code or checkpoints are unavailable, or because the released information is insufficient for faithful reproduction under our evaluation protocol (Zhao et al., 2025; Li et al., 2025b; 2026; Friedmann & Dym, 2026; Chen & Han, 2026; Wang et al., 2026b; Huang et al., 2023). For these methods, we provide qualitative descriptions rather than task-specific parameter settings, focusing on how they reduce, restructure, or replace iterative inference-time generation. For the evaluated baselines, we report the inference parameters that primarily control candidate generation, while the remaining settings follow the original papers or released implementations.

DIFUSCO. DIFUSCO (Sun & Yang, 2023) is a DM-based CO solver that generates candidates through iterative denoising. We use 20 denoising steps at inference time. For each instance, multiple independent initial noises are sampled and separately denoised to obtain candidate solutions. We use 4 initial noise samples by default and 6 on TSP-100. The best solution is retained after decoding and local improvement. Under this setting, DIFUSCO obtains candidate diversity from independent stochastic denoising trajectories, each requiring multiple model evaluations.

Progressive Distillation. Huang et al. (Huang et al., 2023) accelerate pretrained multi-step diffusion solvers through progressive distillation. A student model is trained to approximate multiple teacher denoising transitions using fewer reverse steps, and the trajectory is recursively compressed to reduce the number of model evaluations required for candidate generation.

DISCO. DISCO (Zhao et al., 2025) is an efficient DM-based solver designed for large-scale CO instances. Rather than distilling a pretrained denoising trajectory, it restructures the diffusion process itself. DISCO constrains the sampling space using solution residues, such that generation starts from a mixture of noise and a degraded feasible solution, and further employs an analytically solvable denoising process to reduce the number of reverse-time steps.

Fast T2T. Fast T2T (Li et al., 2024) accelerates diffusion-based CO solving through optimization consistency training. It learns to map noisy states at different noise levels directly toward the solution distribution, enabling few-step or one-step candidate generation. At inference time, Fast T2T

can further apply consistency-based gradient search, where latent solution probabilities are updated under objective guidance and alternated with noise injection and denoising. We set $T_s = 1$ and $T_g = 1$ for all tasks, where T_s denotes the number of sampling steps and T_g the number of gradient-search steps. We use one initial noise sample per instance on TSP-50, TSP-100, RB-[200–300], and ER-[700–800], and two initial noise samples on TSP-500 and TSP-1000.

DEITSP. DEITSP (Wang et al., 2025a) trains a one-step diffusion model with self-consistency enhancement, allowing noisy states at different noise levels to be mapped directly to solution heatmaps. At inference time, it alternates between adding and removing noise and aggregates denoised outputs from different iterations as candidate solutions. The best candidate is retained after decoding and local improvement. We use four sampling runs on both TSP-50 and TSP-100, with four and eight independent initial noises, respectively. For TSP-500 and TSP-1000, we use one initial noise sample per instance.

COExpander. COExpander (Ma et al., 2025a) introduces an adaptive expansion paradigm that progressively constructs solutions from partial decisions. A global predictor produces heatmaps, the most confident variables are fixed, and the resulting partial solution is fed back to guide subsequent predictions. We use $S = 4$, $D_s = 3$, and $I_s = 1$ for all tasks except ER-[700–800], where S denotes the sampling number, D_s the number of determination steps, and I_s the number of inference steps. On ER-[700–800], we use $S = 1$, $D_s = 1$, and $I_s = 1$.

GenSCO. GenSCO (Li et al., 2025b) treats generation as an inference-time search operator rather than a standalone solver. It trains a lightweight rectified-flow model for solution-to-solution refinement. Each inference cycle perturbs the current solution using local search operators, applies the learned flow model to enhance the disrupted solution, and then performs decoding and post-processing. Repeated application of this operator enables inference-time exploration of the learned solution space.

StruDiCO. StruDiCO (Wang et al., 2025b) is a structured diffusion-based CO solver designed to improve the efficiency and reliability of accelerated diffusion inference. It introduces a variable-absorption noising process in which selected variables are progressively deactivated and reconstructed through variable selection in the reverse process. It further employs constrained consistency sampling and gradient-free objective-aware refinement at inference time. We use the same inference settings as Fast T2T, with $T_s = 1$ and $T_g = 1$ for all tasks, where T_s denotes the number of sampling steps and T_g the number of refinement steps. We use one initial noise sample per instance on TSP-50, TSP-100, RB-[200–300], and ER-[700–800], and two initial noise samples on TSP-500 and TSP-1000.

EFLOCO. EFLOCO (Li et al., 2026) formulates solution generation as discrete flow matching. It learns structured transition paths from source configurations to target solutions, introduces a velocity-based consistency loss for stabilizing few-step trajectory evolution, and uses an adaptive time-step scheduler to allocate computation to critical transition regions. In this way, EFLOCO replaces stochastic diffusion dynamics with deterministic flow-matching trajectories for efficient few-step generation.

CycFlow. CycFlow (Friedmann & Dym, 2026) is a deterministic geometric flow-based solver for Euclidean TSP. Rather than generating stochastic heatmaps over the $N \times N$ edge space, it learns an instance-conditioned vector field that transports input coordinates to a canonical circular arrangement. The tour is then recovered by sorting the transported points according to their angular positions.

EDISCO. EDISCO (Chen & Han, 2026) is a geometry-aware diffusion-based solver for geometric CO problems such as TSP. It combines an $E(2)$-equivariant graph neural network with continuous-time categorical diffusion to incorporate rotation, translation, and reflection symmetries into the score model. Its continuous-time formulation supports flexible diffusion inference, while the equivariant architecture improves geometric inductive bias and generalization.

NEXCO. NEXCO (Wang et al., 2026b) makes adaptive expansion native to masked diffusion. It adopts a CO-specific one-way corruption process such that intermediate diffusion states correspond to meaningful partial solutions. A time-agnostic GNN predicts confidence scores for the remaining variables, while a feasibility projection operator selects valid high-confidence variables to progressively expand the partial solution.

DriftCO. For DriftCO, we use 4 latent noise samples on TSP-50, 6 on TSP-100, 6 on RB-[200–300], and 3 on ER-[700–800]. On TSP-500 and TSP-1000, we directly reuse the TSP-100 inference configuration, including 6 latent noise samples per instance. Each noise sample is mapped to a candidate heatmap through a single forward pass, without inference-time denoising or iterative refinement. The final solution is selected from the decoded and post-processed candidates.

F.2 Results on TSPLIB

We further evaluate zero-shot transfer from synthetic TSP instances to TSPLIB benchmark instances. Specifically, all methods use models trained on TSP-100 and are directly evaluated on TSPLIB instances with 50–200 nodes, without TSPLIB-specific training or fine-tuning. Following prior work (Sun & Yang, 2023), we normalize the coordinates of each instance to the range $[0, 1]$ before inference. Table 4 reports the reference gaps of DriftCO and the baselines. DriftCO achieves the lowest average gap of 0.140%, demonstrating effective transfer from synthetic training instances to heterogeneous benchmark instances.

Table 4: Reference gaps (%) on TSPLIB instances with 50–200 nodes. All methods use models trained on TSP-100 without TSPLIB-specific training or fine-tuning.

Instances	DIFUSCO	DEITSP	Fast T2T	StruDiCO	COExpander	**DriftCO (Ours)**
eil51	1.029%	0.201%	0.000%	0.000%	0.201%	0.000%
berlin52	1.249%	0.000%	0.000%	0.372%	1.179%	0.041%
st70	0.172%	0.000%	0.000%	0.000%	0.000%	0.000%
eil76	0.430%	0.218%	0.235%	0.004%	0.004%	0.169%
pr76	0.163%	0.000%	0.000%	0.000%	0.163%	0.000%
rat99	0.016%	0.007%	0.000%	0.000%	0.000%	0.000%
kroA100	0.000%	0.000%	0.000%	0.000%	0.001%	0.000%
kroB100	0.217%	0.569%	0.050%	0.000%	0.401%	0.050%
kroC100	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
kroD100	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
kroE100	0.138%	0.018%	0.000%	0.000%	0.000%	0.018%
rd100	0.000%	0.000%	0.000%	0.000%	0.012%	0.000%
eil101	0.840%	0.661%	0.000%	0.000%	0.021%	0.000%
lin105	0.825%	0.289%	0.000%	0.107%	0.057%	0.057%
pr107	2.100%	0.353%	0.589%	1.380%	0.513%	0.000%
pr124	3.838%	0.000%	0.809%	0.441%	1.075%	0.584%
bier127	0.508%	0.344%	0.238%	0.682%	0.986%	0.434%
ch130	0.156%	0.269%	0.000%	0.000%	0.000%	0.058%
pr136	0.124%	0.295%	0.000%	0.000%	0.000%	0.240%
pr144	1.958%	0.019%	0.102%	0.051%	0.069%	0.000%
ch150	0.657%	0.275%	0.487%	0.657%	0.115%	0.272%
kroA150	1.359%	0.196%	0.000%	0.000%	0.000%	0.193%
kroB150	0.430%	0.390%	0.000%	0.334%	0.000%	0.324%
pr152	2.075%	0.563%	1.376%	0.687%	1.981%	0.635%
u159	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
rat195	2.521%	0.801%	0.770%	1.965%	0.189%	0.888%
d198	0.000%	0.000%	0.000%	0.000%	0.003%	0.000%
kroA200	1.204%	0.436%	0.530%	0.171%	0.288%	0.111%
kroB200	0.872%	0.971%	0.000%	0.012%	0.012%	0.000%
Mean	0.789%	0.237%	0.179%	0.237%	0.251%	**0.140%**

F.3 Ablation Studies

Effectiveness of attraction–repulsion learning. To isolate the contribution of the drifting objective from standard supervised learning, we compare DriftCO with two one-step generators using the same architecture and exactly the same single reference solution for each training instance. The first is trained with binary cross-entropy (BCE) against the reference solution. The second removes the repulsion term from DriftCO and retains only single-reference attraction. Since $\mathbf{V}^{+}(\hat{\mathbf{x}}; G) = \mathbf{y}_G^* - \hat{\mathbf{x}}$, the attraction-only variant reduces exactly to direct L_2 regression toward the reference solution. Thus, all variants share the same architecture, supervision, and inference procedure, differing only in the training objective.

As shown in Table 5, DriftCO consistently achieves the lowest reference gap across all evaluated settings. On TSP-500 and TSP-1000, removing the repulsion term substantially degrades performance, while BCE yields even larger gaps. On MIS, DriftCO also outperforms both alternatives on the in-distribution RB-[200–300] and zero-shot ER-[700–800] settings. These results show that the improvement of DriftCO cannot be attributed to the network architecture or single-reference supervision alone, and demonstrate the benefit of same-instance repulsion beyond pointwise supervision.

Table 5: Ablation of the training objective. Results are reported as reference gap (%, lower is better).

Method	**TSP-500**	**TSP-1000**	**RB-[200–300]**	**ER-[700–800]**
Supervised (BCE)	5.38	5.44	4.17	12.06
Attraction-only (L_2)	2.99	3.96	4.42	11.99
DriftCO	**2.23**	**3.24**	**2.63**	**10.87**

Effectiveness of the structural cosine kernel. Table 6 compares the proposed structural cosine kernel with Gaussian (Lai et al., 2026) and Laplacian kernels (Deng et al., 2026). The structural cosine kernel consistently achieves the lowest reference gap across TSP and MIS. In contrast, both Euclidean-distance-based kernels yield larger gaps, indicating that Euclidean distance between heatmaps is less effective at capturing structural redundancy among combinatorial candidates. These results support our use of structural cosine similarity to measure candidate similarity and weight same-instance repulsion.

Table 6: Ablation of the similarity kernel across TSP and MIS. Results are reported as reference gap (%, lower is better).

Kernel	**TSP-500**	**TSP-1000**	**RB-[200–300]**	**ER-[700–800]**
Gaussian	2.98	4.65	4.37	12.56
Laplacian	3.05	3.44	4.02	12.68
Structural Cosine	**2.23**	**3.24**	**2.63**	**10.87**