EDISCO: Equivariant DIScrete Diffusion for Euclidean Combinatorial Optimization

Ruogu Chen*
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, AB, Canada T6G 1H9
ruogu@ualberta.ca

Jie Han
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, AB, Canada T6G 1H9
jhan8@ualberta.ca

Abstract

Euclidean combinatorial optimization problems (ECOPs), such as the Traveling Salesman Problem (TSP) and Capacitated Vehicle Routing Problem (CVRP), possess inherent symmetries under the two-dimensional Euclidean group E(2), including rotations, reflections, and translations. Existing learning-based methods, including recent diffusion-based methods, rely on data augmentation or regularization to approximate E(2)-equivariance. This paper presents EDISCO, the first discrete diffusion model for ECOPs with exact E(2)-invariant generative distributions over node-index solutions. EDISCO combines an E(2)-equivariant network that predicts edge-selection probabilities with a categorical continuous-time Markov chain over discrete edge variables. Exact posterior sampling supports efficient multi-step inference. The equivariant network gives EDISCO a local geometric inductive bias: edge neighborhoods with the same relative geometry and combinatorial context are represented consistently regardless of absolute position or orientation, making learning more efficient and inference more robust than non-equivariant methods. EDISCO outperforms previous learning-based state-of-the-art solvers on synthetic TSPs from 100 to 10000 nodes and CVRPs from 50 to 2000 customers, while using only 33–50% of the training instances. Trained only on uniform synthetic data, EDISCO also outperforms competing learning-based baselines under spatial distribution shift and CVRP constraint-tightness shift.

1 Introduction

Euclidean combinatorial optimization problems (ECOPs), such as the Traveling Salesman Problem (TSP) and Capacitated Vehicle Routing Problem (CVRP), are core benchmarks for combinatorial optimization (CO) over spatial data. Classical solvers like Concorde and the LKH heuristic have been refined for decades [2, 31], while learning-based solvers have emerged as fast approximate alternatives with amortized inference and instance-level generalization [41, 36]. Diffusion-based methods now lead the non-autoregressive family, pushing optimality gaps below one percent on synthetic benchmarks and scaling ECOP solving from hundreds to ten thousand nodes [78, 48, 94, 49, 84, 50, 85, 59]. The field is now moving beyond chasing the last fraction of a percent on uniform synthetic benchmarks. Practical scenarios require solvers to remain reliable when the spatial distribution, problem size, or constraints shift away from training [7, 58, 8], and robustness to such shifts is now a central criterion for whether learned solvers transfer beyond synthetic benchmarks.

A key obstacle is that ECOP solvers must learn reusable Euclidean structures, such as local distances, neighborhoods, edge crossings, depot-relative geometry, and capacity-pressure patterns. If a model recognizes these structures only in a particular coordinate frame, it must relearn the same patterns across rotated, reflected, and translated copies. The two-dimensional Euclidean group E(2) captures

*Code is available at https://github.com/ValleyC/EDISCO.

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

exactly these irrelevant coordinate dimensions. Rotations, reflections, and translations change the coordinate representation but not the node-index solution [66, 12]. A reliable ECOP model should therefore learn geometric patterns that are constant on E(2) orbits, so the same local pattern induces the same decision wherever and however it appears in the plane. Existing neural CO methods approximate this principle through augmentation or symmetry regularization [39]. However, these mechanisms do not enforce exact equivariance or invariance by construction [24, 64, 11]. A diffusion model defines a generative distribution over solutions rather than a single deterministic prediction. Therefore, for diffusion-based CO, E(2) invariance must hold at the level of the entire distribution rather than only at the predicted output. Achieving this requires every stage of the pipeline (the diffusion model, reverse sampler, and decoder) to preserve the same symmetry. Existing diffusion solvers improve denoising quality, inference speed, local search, and expansion, but leave E(2) symmetry underexplored and do not guarantee exact E(2)-invariance of the generated solution distribution.

This work proposes EDISCO, the first discrete diffusion model for ECOPs with an end-to-end E(2)-invariant generative distribution over node-index solutions. EDISCO introduces an E(2)-equivariant edge-score network coupled with a categorical continuous-time Markov chain (CTMC) [14] over discrete edge variables, so the reverse diffusion process preserves geometric symmetry at the distribution level. Exact CTMC posterior sampling adapts multi-step solvers to discrete combinatorial variables, giving a controllable quality–speed trade-off without retraining. To preserve E(2)-invariance through the final decoding step, EDISCO uses an invariant edge-expansion decoder that reads only edge probabilities, pairwise distances, and combinatorial feasibility constraints. The resulting model represents local edge neighborhoods with the same relative geometry consistently across positions and orientations, which makes learning more efficient and inference more robust than non-equivariant methods.

The contributions are as follows.

1. *Exact distribution-level E(2) invariance for generative ECOP solving.* EDISCO realizes orbit-shared local geometric structure by construction: scalar edge and message channels are constant on E(2) orbits, so equivalent local edge neighborhoods are represented consistently under E(2) transformations. The resulting generative distribution over node-index solutions is exactly E(2)-invariant. Proposition 1 formalizes this induced quotient-space reduction.
2. *Efficient continuous-time categorical diffusion for edge-valued CO.* EDISCO formulates edge selection as a categorical CTMC and uses exact posterior sampling to adapt multi-step solvers to discrete combinatorial variables. This provides flexible quality–speed trade-offs without retraining.
3. *Robust symmetry-preserving ECOP solving under spatial and constraint shifts.* The same invariance principle is preserved under sparse graph construction and capacity-conditioned CVRP features. As a result, EDISCO transfers from uniform synthetic training to spatial out-of-distribution (OOD) benchmarks and capacity-shifted CVRP while maintaining the same geometric symmetry guarantee.

EDISCO achieves leading optimality-gap results among the considered learning-based solvers on TSPs from 100 to 10000 nodes and on CVRPs from 50 to 2000 customers, while using only 33% to 50% of the training instances compared to competing methods. With the model only trained on synthetic data, EDISCO outperforms non-equivariant baselines under spatial distribution shift [7] and achieves the lowest mean optimality gap among the considered learning-based methods on the evaluated TSPLIB instances [72]. On CVRPs, the same invariant-feature principle supports capacity-conditioned routing under the protocol of Luo et al. [58], where a single pre-trained model spans a range of capacity ratios without coordinate-frame dependence. Additional experiments on the Euclidean Steiner Tree Problem (ESTP) and the Maximum Independent Set (MIS) demonstrate EDISCO's generality.

2 Related Work

Neural and diffusion-based ECOP solvers. Autoregressive solvers such as AM [41], POMO [43], and BQ-NCO [20] build tours sequentially, while heatmap methods predict pairwise edge structure and recover tours through search or local construction [35, 25, 86]. Discrete diffusion [3] extends this to generative modeling of edge variables, refined by DIFUSCO [78], T2T [48], Fast-T2T [49], CADO [91], and DISCO [94]. Expansion solvers add local construction on top of global prediction,

including COExpander [59] and the diffusion-expansion hybrids StruDiCO [84], GenSCO [50], and NEXCO [85]. Across these methods, geometric symmetry is approached through orbit-sampling rather than architectural orbit-sharing, providing no exact distribution-level invariance guarantee.

Equivariance and geometric structure. Equivariant architectures enforce equivariance by construction [16, 12, 75], applying a single scalar feature map across every E(2)-equivalent input, a property referred to here as *orbit-sharing*. This restricts invariant functions to the quotient space and can improve data efficiency. Augmentation and symmetry regularization approximate the same effect through *orbit-sampling*, without enforcing exact equivariance by construction [11, 24, 64]. In neural CO, Sym-NCO [39] samples orbits via a symmetry regularizer, while geometric models exploit Euclidean structure for TSP generalization [66, 76]. Symmetry-aware diffusion has also been developed for molecular generation with denoising equivariant under the three-dimensional Euclidean group E(3) [33] and for crystal structure prediction with lattice-permutation and periodic-translation symmetries [52].

Continuous-time categorical diffusion and decoding. Continuous-time categorical diffusion admits closed-form transitions and flexible inference schedules [14, 77]. Diffusion-based CO solvers recover discrete solutions through greedy extraction, local search, or expansion, with NEXCO [85] making expansion native to diffusion.

Constraint-aware and realistic routing. Neural solvers are increasingly evaluated beyond uniform synthetic TSP, including TSPLIB transfer [72], cross-distribution routing [7], real-world hierarchical TSP [28], and generalist CO agents [21]. For vehicle routing, RouteFinder [6], CaDA [47], Luo et al. [58], and CaR [8] target constraint-aware or capacity-robust routing.

3 Method

3.1 Problem Formulation

An ECOP instance has n nodes with coordinates $\{\mathbf{c}_i\}_{i=1}^n \subset \mathbb{R}^2$, and asks for a binary decision matrix $\mathbf{X} \in \{0,1\}^{n\times n}$ that minimizes a distance-based cost f subject to problem-specific constraints $\mathcal{C}$:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} f(\mathbf{X}, \{\mathbf{c}_i\}_{i=1}^n) \quad \text{s.t. } \mathbf{X} \in \mathcal{C}. \tag{1}$$

For TSPs, $\mathbf{X}$ is the tour adjacency, f is the total edge length, and $\mathcal{C}$ encodes Hamiltonicity. EDISCO learns the conditional distribution $p(\mathbf{X} \mid \{\mathbf{c}_i\}_{i=1}^n)$ through a generative process [78, 48].

3.2 Categorical CTMC Diffusion over Edge Variables

EDISCO models edge selection as a categorical diffusion process directly over the binary variables in $\mathbf{X}$, avoiding continuous relaxations of discrete tour structure and exposing closed-form transition probabilities for training and accelerated sampling [3, 14].

Forward process The CTMC corrupts the clean solution $\mathbf{X}_0$ with instantaneous transition rate

$$\mathbf{Q}(t) = \beta(t)\left(\mathbb{1}\mathbb{1}^T - K\mathbf{I}\right), \tag{2}$$

where $K = 2$ for binary edges, $\mathbb{1} \in \mathbb{R}^K$ is the all-ones vector, and $\mathbf{I} \in \mathbb{R}^{K\times K}$ is the identity matrix, with linear schedule $\beta(t) = \beta_{\min} + t(\beta_{\max} - \beta_{\min})$, $t \in [0,1]$, $\beta_{\min} = 0.1$, $\beta_{\max} = 1.5$ (alternatives are evaluated in Appendix F.5). The Kolmogorov forward equation [65] admits the closed-form transition probability [14]

$$P_{ab}(t|s) = \frac{1}{K} + \left(\delta_{ab} - \frac{1}{K}\right)\exp\left(-K\int_s^t \beta(u)du\right), \tag{3}$$

where $a, b \in \{0, \ldots, K-1\}$ are the indices of the categorical states. Specializing to binary edge variables ($K = 2$, $a, b \in \{0,1\}$),

$$P(X_{t,ij} = b \mid X_{0,ij} = a) = P_{ab}(t|0) = \frac{1}{2} + \left(\delta_{ab} - \frac{1}{2}\right)e^{-2\int_0^t \beta(u)du}, \tag{4}$$

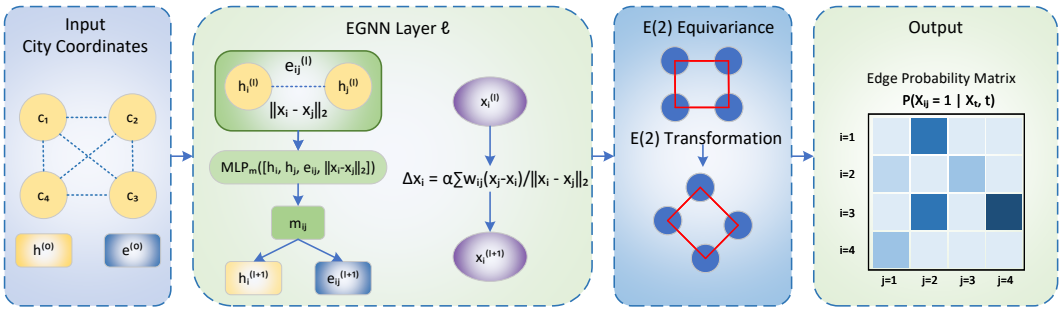

Figure 1: EDISCO's E(n)-equivariant graph neural network (EGNN) architecture. EGNN layers update coordinates equivariantly and output edge probabilities that remain invariant under rotations, reflections, and translations.

with the linear schedule giving the analytic integral

$$\int_0^t \beta(u)du = \beta_{\min}t + \frac{1}{2}(\beta_{\max} - \beta_{\min})t^2, \tag{5}$$

so $\mathbf{X}_t$ can be sampled exactly from $\mathbf{X}_0$ at any time without simulating intermediate jumps.

Learned reverse process Given $\mathbf{X}_0$, the exact one-step posterior follows from Bayes' rule,

$$q(\mathbf{X}_{t-\Delta t}|\mathbf{X}_t, \mathbf{X}_0) = \frac{q(\mathbf{X}_t|\mathbf{X}_{t-\Delta t})\, q(\mathbf{X}_{t-\Delta t}|\mathbf{X}_0)}{q(\mathbf{X}_t|\mathbf{X}_0)}. \tag{6}$$

However, $\mathbf{X}_0$ is unknown at inference. The score network instead predicts $\hat{p}_\theta(\mathbf{X}_0|\mathbf{X}_t, t, \{\mathbf{c}_i\})$, and the learned reverse kernel marginalizes Equation 6 over this predicted clean-edge distribution. This x_0 parameterization remains stable near low-noise timesteps where reconstruction accuracy matters most [73].

3.3 E(2)-Equivariant Score Network

Rotations, reflections, and translations of an ECOP coordinate frame leave the node-index solution unchanged, so EDISCO predicts edge variables with an E(n)-equivariant graph neural network (EGNN) [75] that propagates coordinate transformations equivariantly while keeping edge logits invariant. Figure 1 illustrates EDISCO's equivariant score-network architecture. The score network predicts clean-edge probabilities conditioned on the noisy edge state, diffusion time, and instance coordinates.

The network maintains invariant node features $\mathbf{h}_i$, invariant edge features $\mathbf{e}_{ij}$, and equivariant coordinates $\mathbf{x}_i$. Pairwise messages, computed by a multilayer perceptron (MLP), use only invariant scalar geometry:

$$\mathbf{m}_{ij}^{(\ell)} = \text{MLP}_m\left([\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)}, \mathbf{e}_{ij}^{(\ell)}, \|\mathbf{x}_i^{(\ell)} - \mathbf{x}_j^{(\ell)}\|_2]\right). \tag{7}$$

Coordinate updates are constrained to scalar-weighted relative vectors,

$$\Delta\mathbf{x}_i = \alpha \sum_{j\neq i} w_{ij} \cdot \frac{\mathbf{x}_j^{(\ell)} - \mathbf{x}_i^{(\ell)}}{\|\mathbf{x}_j^{(\ell)} - \mathbf{x}_i^{(\ell)}\|_2 + \epsilon}, \tag{8}$$

with $w_{ij} = \tanh(\text{MLP}_c(\mathbf{m}_{ij}^{(\ell)})/\tau)$, $\tau = 10$, $\alpha = 0.1$. Because w_{ij} is built from invariant messages, the update transforms covariantly with the coordinates. Edge and node states then update as

$$\mathbf{e}_{ij}^{(\ell+1)} = \text{LN}\big((\mathbf{e}_{ij}^{(\ell)} + \text{MLP}_e([\mathbf{e}_{ij}^{(\ell)}, \mathbf{m}_{ij}^{(\ell)}])) \odot (1 + \text{MLP}_t(\mathbf{t}_{\text{emb}}))\big), \tag{9}$$

$$\mathbf{h}_i^{(\ell+1)} = \text{LN}\big(\mathbf{h}_i^{(\ell)} + \text{MLP}_h([\mathbf{h}_i^{(\ell)}, \textstyle\sum_{j\neq i} \mathbf{m}_{ij}^{(\ell)}])\big), \tag{10}$$

where LN denotes layer normalization and $\mathbf{t}_{\text{emb}}$ is a sinusoidal time embedding [79]. Because the output head reads only invariant edge states, the predicted distribution over node-index edge variables is invariant under E(2), and Corollary 2 formalizes the orbit-shared scalar representation that follows. For large instances, EDISCO restricts message passing to an invariant k-nearest-neighbor (kNN) graph constructed from pairwise distances, with deterministic tie-breaking by node indices, so sparsification preserves the invariance argument.

Proposition 1. *Let $X = \mathbb{R}^{2n}$ denote the space of ordered two-dimensional coordinates for n nodes, and let $G = \mathrm{E}(2)$ be the Euclidean group acting on X by simultaneous rotation, reflection, and translation of all node positions. Let $X_{\mathrm{gp}} \subset X$ be the subset of configurations in general position (pairwise distinct points with no three collinear), and assume $n \geq 3$. Then:*

(i) The quotient space X_{gp}/G is a smooth manifold of dimension $2n - 3$.

(ii) Any G-invariant function $F : X_{\mathrm{gp}} \to Y$ factors uniquely through the quotient as $F = \widetilde{F} \circ \pi$, where $\pi : X_{\mathrm{gp}} \to X_{\mathrm{gp}}/G$ is the canonical projection and $\widetilde{F} : X_{\mathrm{gp}}/G \to Y$ is a function on the quotient manifold.

(iii) Learning a G-invariant function is equivalent to learning a function on the $(2n - 3)$-dimensional manifold X_{gp}/G rather than on $\mathbb{R}^{2n}$.

Corollary 2 (Orbit-shared scalar representations)**.** *Let $\mathbf{x} = (\mathbf{c}_1, \ldots, \mathbf{c}_n)$ denote the input configuration and let $g \in \mathrm{E}(2)$ act as $g \cdot \mathbf{x} = (g \cdot \mathbf{c}_1, \ldots, g \cdot \mathbf{c}_n)$. Then at every EGNN layer ℓ, the scalar features satisfy*

$$\mathbf{m}_{ij}^{(\ell)}(g \cdot \mathbf{x}) = \mathbf{m}_{ij}^{(\ell)}(\mathbf{x}), \quad \mathbf{e}_{ij}^{(\ell)}(g \cdot \mathbf{x}) = \mathbf{e}_{ij}^{(\ell)}(\mathbf{x}), \quad \mathbf{h}_i^{(\ell)}(g \cdot \mathbf{x}) = \mathbf{h}_i^{(\ell)}(\mathbf{x}),$$

while the coordinate features transform equivariantly: $\mathbf{x}_i^{(\ell)}(g \cdot \mathbf{x}) = g \cdot \mathbf{x}_i^{(\ell)}(\mathbf{x})$. Consequently, any input configuration $\mathbf{x}$ and every element of its E(2)-orbit produce the same scalar feature field at every layer, so a single learned scalar feature map applies uniformly across the orbit.

The quotient dimension $2n - 3$ in Proposition 1 reflects E(2)'s three continuous degrees of freedom, namely two translations and one rotation, with reflections contributing only a discrete subgroup. The full proof and the derivation that this invariance is preserved through diffusion and decoding are in Appendices B.1 and B.2.

Capacity conditioning for CVRPs. For CVRPs, capacity information enters the score network only through E(2)-invariant scalar channels. With vehicle capacity Q, default capacity Q_{default}, ratio $\lambda = Q/Q_{\mathrm{default}}$, and demands d_i, each node receives d_i/Q, each edge receives $(d_i + d_j)/Q$ and $|d_i - d_j|/Q$, and the global input $[\log \lambda, \sum_i d_i/Q]$ is embedded as $\mathbf{z}_c$. This embedding modulates only scalar message channels via FiLM [68],

$$\tilde{\mathbf{m}}_{ij}^{(\ell)} = \boldsymbol{\gamma}_\ell(\mathbf{z}_c) \odot \mathbf{m}_{ij}^{(\ell)} + \boldsymbol{\beta}_\ell(\mathbf{z}_c), \tag{11}$$

with $\boldsymbol{\gamma}_\ell$ and $\boldsymbol{\beta}_\ell$ produced by lightweight MLPs initialized near identity. Because every conditioning input is invariant and the coordinate update of Equation 8 is unchanged, the conditional distribution remains E(2)-invariant for every fixed capacity. Section 4.3 evaluates this variant under constraint shift.

Training minimizes cross-entropy on x_0 prediction with signal-to-noise ratio (SNR)-motivated weighting (Algorithm 1). Here, s_θ is the score network with parameter θ, $\mathcal{L}$ is the weighted loss, and CE in Algorithm 1 denotes edge-wise cross-entropy. AdamW updates θ using $\nabla_\theta \mathcal{L}$, the loss gradient. For CVRPs, each sampled instance supplies demands d_i and capacity Q to the decoder. Capacity conditioning $\mathbf{z}_c$ and these inputs are omitted for TSPs. At each step a random time t and clean instances are sampled, the noisy state is drawn from the closed-form forward process, and the $(1 - \sqrt{t})$ weighting emphasizes low-noise timesteps where reconstruction targets are reliable [40, 30]. Full training configuration is in Appendix E.6.

3.4 Multi-Step CTMC Sampling

The continuous-time formulation allows the inference schedule to be changed without retraining. EDISCO adapts multi-step solvers such as PNDM [55] and DEIS [93] to categorical CTMCs by separating prediction smoothing from discrete posterior sampling. At each reverse step, the score network predicts $\hat{p}_\theta(\mathbf{X}_0 \mid \mathbf{X}_t, t, \{\mathbf{c}_i\})$, and Adams-Bashforth (AB)-style smoothing combines recent predictions into $\tilde{p}$. Smoothing starts with the current prediction alone, uses a lower order until its history fills, and clamps probabilities to $[0, 1]$. For each edge (i, j), sampling then uses the explicit categorical kernel

$$q(x_{s,ij} \mid x_{t,ij}, \tilde{p}_{ij}) = \sum_{x_0 \in \{0,1\}} \frac{P(x_{t,ij} \mid x_{s,ij})\, P(x_{s,ij} \mid x_0)}{P(x_{t,ij} \mid x_0)}\, \tilde{p}_{ij}(x_0 \mid x_{t,ij}), \tag{12}$$

where $s < t$, $\tilde{p}_{ij}(x_0 \mid x_{t,ij})$ is the smoothed probability of clean state x_0, and the transition probabilities $P(\cdot \mid \cdot)$ follow from Equation 3. Edges are sampled independently from this kernel to form $\mathbf{X}_s$. This preserves the categorical transition structure while giving a controllable quality–speed trade-off. Appendix F.1 details prediction smoothing and posterior marginalization, and Appendix F.2 evaluates solver choices.

3.5 Native Edge Expansion (NEE)

Denoising outputs an edge-probability matrix $P \in [0,1]^{n \times n}$ with $P_{ij} = p(X_{ij} = 1)$. NEE converts this invariant score field into a feasible tour without external local-search refinement. For TSPs, undirected candidates are ranked by

$$r_{ij} = \frac{P_{ij} + P_{ji}}{2(d_{ij} + \epsilon)}, \qquad d_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\|_2, \tag{13}$$

and the highest-ranked feasible edge is added iteratively. The feasibility projection maintains node degrees, a partial edge set S_t, and a union-find structure, accepting an edge only when both endpoints have degrees below two and the edge does not close a premature subtour, with the final Hamiltonian closure as the only allowed cycle-closing step. For CVRPs, customer nodes keep degree two while the depot accepts one edge pair per active route. The projection tracks per-route accumulated demand, and a candidate edge that would push any active route's demand above Q is rejected. The matching depot-return edge closes that partial route before the next candidate is evaluated. All inputs used by the decoder (pairwise distances, edge probabilities, capacity-derived features, partial edge sets, union-find connectivity) and all feasibility checks are invariant under E(2), so the selected node-index edge set is identical under any $g \in \mathrm{E}(2)$, with the full preservation argument in Appendix B.2.

Algorithm 2 initializes $\mathbf{X}_{t_0}$ from the CTMC equilibrium, with the residual mismatch absorbed within the first denoising steps. The schedule $\{t_k\}_{k=0}^{T}$ contains T reverse steps from $t_0 = 1$ to $t_T = 0$, followed by a final clean-state prediction at $t_T = 0$. Greedy extraction and 2-opt local search [17] remain reference baselines, but NEE is the native score-to-tour decoder whose invariance follows from its inputs and feasibility projection.

Algorithm 1 EDISCO Training

Require: Dataset $\mathcal{D}$, score network s_θ
Require: CVRP only: d_i, Q, Q_{default}
while not converged **do**
  $(\mathbf{X}_0, \{\mathbf{c}_i\}) \sim \mathcal{D}, \quad t \sim \mathcal{U}(0,1)$
  $\mathbf{z}_c \leftarrow \mathrm{MLP}([\log \lambda, \sum_i d_i/Q])$ if CVRP
  $\mathbf{X}_t \sim q(\mathbf{X}_t|\mathbf{X}_0)$ (Eq. 4)
  $\hat{\mathbf{X}}_0 \leftarrow s_\theta(\mathbf{X}_t, t, \{\mathbf{c}_i\}, \mathbf{z}_c)$
  $\mathcal{L} \leftarrow (1 - \sqrt{t}) \cdot \mathrm{CE}(\hat{\mathbf{X}}_0, \mathbf{X}_0)$
  $\theta \leftarrow \mathrm{AdamW}(\theta, \nabla_\theta \mathcal{L})$
end while

Algorithm 2 EDISCO Inference

Require: $\{\mathbf{c}_i\}$, trained score network s_θ
Require: Schedule $\{t_k\}_{k=0}^{T}$
Require: CVRP only: d_i, Q, Q_{default}
$\mathbf{X}_{t_0} \sim \mathrm{Uniform}(\{0,1\}^{n \times n})$
$\mathbf{z}_c \leftarrow \mathrm{MLP}([\log \lambda, \sum_i d_i/Q])$ if CVRP
for $k = 0$ to $T - 1$ **do**
  $\hat{\mathbf{X}}_0 \leftarrow s_\theta(\mathbf{X}_{t_k}, t_k, \{\mathbf{c}_i\}, \mathbf{z}_c)$
  $\tilde{p} \leftarrow \mathrm{ABStyleSmooth}(\{\hat{\mathbf{X}}_0^{(t_j)}\}_{j \le k})$
  $\mathbf{X}_{t_{k+1}} \sim q(\cdot|\mathbf{X}_{t_k}, \tilde{p})$ (Eq. 12)
end for
$\hat{\mathbf{X}}_0 \leftarrow s_\theta(\mathbf{X}_{t_T}, t_T, \{\mathbf{c}_i\}, \mathbf{z}_c)$
return $\mathrm{NEE}(\hat{\mathbf{X}}_0, \{\mathbf{c}_i\}, \{d_i\}, Q)$

4 Experiments

We use N for the number of TSP cities or CVRP customers. Main results cover TSP-100/500/1000 and CVRP-50/100/200/500, with TSP-10000 and CVRP-1000/2000 extended in Appendices F.3 and A.2. ESTP (Appendix A.1) and MIS (Appendix A.3) test Euclidean tree construction and engine generality on a non-Euclidean problem.

4.1 Setup

Datasets. TSPs follow Kool et al. [41] with coordinates uniform on $[0,1]^2$, Concorde [2] labels at TSP-100, LKH-3 [31] labels at TSP-500/1000/10000, and the standard 1,280-instance and 128-instance test sets from Kool et al. [41] and Fu et al. [25]. EDISCO uses only 33–50% of the training data of competing diffusion methods at every TSP and CVRP scale (Appendices F.7 and E.6). CVRP sizes, capacities, and test-set counts follow Ma et al. [59] (Appendix H.2). We use integer demands

$\{1, \ldots, 9\}$, capacities $Q \in \{40, 50, 80, 100\}$ for $N \in \{50, 100, 200, 500\}$, HGS-CVRP labels [80], and 10K/10K/100/100 test instances at $N \in \{50, 100, 200, 500\}$, with gaps reported against HGS-CVRP since exact optima are unavailable. The capacity-shift evaluation (Section 4.3) follows Luo et al. [58].

Graph representation. Dense adjacency is used for $N \leq 100$ and kNN sparsification [78] at larger scales. Because neighbor sets depend only on pairwise Euclidean distances, sparsification itself is E(2)-invariant, so the invariance argument carries over unchanged. Memory drops from $O(n^2)$ to $O(kn)$, enabling training and inference on a single 48 GB GPU at all reported scales (Appendix F.8).

4.2 TSP Results

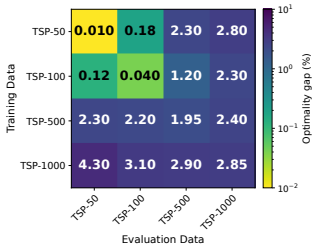

Figure 2: Cross-size generalization for TSP models trained at one scale and evaluated across all scales under greedy decoding. Values are optimality gaps (%).

Table 1 reports TSPs with $N \in \{100, 500, 1000\}$ with canonical configurations following Li et al. [50], with each method's decoding configuration listed in the table. EDISCO uses 5-step DEIS-2 [93] with NEE. TSP-10000, the 50-step PNDM [55] quality variant, sampling-based decoding, and TSPLIB are in Appendices F.3 and F.4.

TSP across scales. EDISCO leads on Gap at all three scales and is the fastest among modern diffusion- and expansion-based solvers. Faster non-generative baselines such as POMO exist but at much higher Gap (above 22% at TSP-500/1000). EDISCO also leads both Gap and Time among the evaluated learning-based solvers with sub-2% Gap at TSP-10000 (Appendix F.3), with compute and runtime conventions in Appendix F.10. The TSP-1000-trained model in Figure 2 has gaps of at most 4.3% across all other scales, exceeding the cross-size generalization of DIFUSCO [78] and T2T [48].

Cross-distribution generalization. Following Bi et al. [7], we evaluate on Uniform together with three OOD variants Cluster, Explosion, and Implosion (Table 2, Det.(%) $= (\text{Gap}_{\text{OOD}}/\text{Gap}_{\text{Uniform}} - 1) \times 100$). EDISCO's average deterioration is 4% against 15% (GLOP), 133% (DIFUSCO), 687% (T2T), and 1961% (Fast-T2T), consistent with Corollary 2: orbit-shared scalar features depend only on local geometry rather than on coordinate-frame patterns of the training distribution. Tour visualizations are in Figure 3 and Appendix F.9.

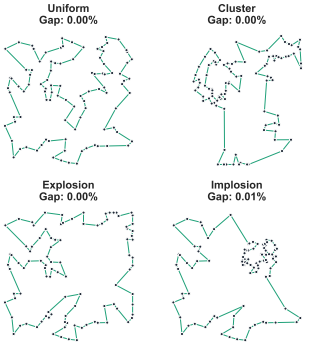

Figure 3: EDISCO tours on Uniform and three OOD distributions for TSP-100.

4.3 CVRP Results

Table 3 reports CVRPs with $N \in \{50, 100, 200, 500\}$ against HGS-CVRP. EDISCO extends the NEE decoder in Table 1 with capacity-feasibility checks and capacity-conditioned invariant features (Section 3.3), preserving E(2)-invariance through the constrained-routing pipeline.

EDISCO leads on Gap at all four scales and removes the external local-search dependency, with Time fastest at CVRP-500 and within 2× of COExpander+LS at smaller scales. Larger CVRPs with $N \in \{1000, 2000\}$ use a divide-and-conquer extension (Appendix A.2).

Constraint robustness on CVRPs. Following Luo et al. [58], CVRP-100 is evaluated at seven absolute capacities $Q \in \{10, 50, 100, 200, 300, 400, 500\}$ spanning tight (∼2 customers per route) to near-TSP regimes, with 10,000 instances per bin. Four EDISCO variants isolate conditioning: *default-only* (trained at Q=50), *mixed unconditioned* (per-instance $Q \sim \text{Unif}\{10, \ldots, 500\}$, no conditioning input), *capacity-conditioned* (proposed, same distribution with invariant features and modulated invariant messages), and *per-capacity specialists* (upper bound). Neural baselines AM, POMO, Luo's multi-expert solver [58], CaR [8], StruDiCO, NEXCO, and COExpander cover the full sweep, the last three trained at Q=50.

Table 1: TSP results at $N \in \{100, 500, 1000\}$. Gap is relative to Concorde (TSP-100) or LKH-3 (TSP-500/1000). Time is per-instance wall-clock at batch size 1, single-thread [84, 85], with hardware in Appendix E.2. Throughout the tables, ↓ indicates lower is better and ↑ indicates higher is better. **Best** and second-best per Gap and Time among modern diffusion- and expansion-based solvers (COExpander and Generative Neural Solvers). RL: reinforcement learning, SL: supervised learning, G: greedy, BS: beam search, 2O: 2-opt, S: sampling, AE/NAE/NEE: adaptive / native adaptive / native edge expansion.

Method	Type	TSP-100			TSP-500			TSP-1000		
		Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓
Reference Solvers										
Concorde [2]	Exact	7.76	0.00%	0.23 s	16.55	0.00%	18.65 s	23.12	0.00%	84.41 s
LKH-3 [31]	Heuristic	7.76	0.00%	0.12 s	16.55	0.00%	4.64 s	23.12	0.00%	17.54 s
Non-Generative Neural Solvers										
AM [41]	RL+BS	7.95	2.48%	0.61 s	19.53	18.03%	1.31 s	29.90	29.24%	5.91 s
POMO [43]	RL+×8	7.77	0.13%	0.006 s	20.22	22.19%	0.47 s	32.55	40.57%	3.75 s
GCN [35]	SL+BS	8.41	8.38%	0.28 s	30.37	83.55%	17.81 s	51.26	121.73%	24.23 s
DIMES [71]	RL+G	8.01	3.23%	0.06 s	17.17	3.74%	0.45 s	24.79	7.22%	1.12 s
BQ-NCO [20]	SL+BS	7.76	0.01%	0.19 s	16.64	0.55%	7.03 s	23.47	1.38%	57.81 s
GLOP [90]	RL+G	–	–	–	17.10	3.32%	0.17 s	23.84	3.11%	1.40 s
COExpander [59]	SL+AE+2O	7.76	0.04%	0.18 s	16.63	0.49%	0.61 s	23.34	0.95%	2.26 s
Generative Neural Solvers										
DIFUSCO [78]	SL+G+2O	7.78	0.22%	0.59 s	16.81	1.55%	1.43 s	23.55	1.86%	5.04 s
T2T [48]	SL+G+2O	7.76	0.06%	1.13 s	16.68	0.78%	3.05 s	23.41	1.25%	9.23 s
Fast-T2T [49]	SL+G+2O	7.76	0.03%	1.10 s	16.61	0.39%	1.39 s	23.25	0.58%	5.78 s
CADO [91]	SL+RL+G+2O	7.76	0.01%	0.75 s	16.59	0.24%	2.50 s	23.28	0.69%	8.50 s
DISCO [94]	SL+G	7.80	0.58%	0.50 s	–	–	–	–	–	–
StruDiCO [84]	SL+G+2O	7.76	0.024%	0.083 s	16.60	0.326%	0.674 s	23.24	0.535%	2.614 s
GenSCO [50]	SL (C=10)	7.76	**0.000%**	0.50 s	16.55	0.023%	1.10 s	23.25	0.559%	3.50 s
NEXCO [85]	SL+NAE	7.76	0.03%	0.08 s	16.59	0.28%	0.33 s	23.26	0.63%	1.31 s
EDISCO (this work)	SL+NEE	7.76	**0.000%**	**0.075 s**	16.55	**0.018%**	**0.28 s**	23.24	**0.52%**	**1.28 s**

Table 2: Cross-distribution generalization on TSP-100 [7], EDISCO and diffusion baselines trained on Uniform with greedy + 2-opt decoding. EDISCO's SL+NEE configuration appears in Table 1. Det.(%): deterioration relative to Uniform. Average Gap includes Uniform and the three OOD variants, while average Det. covers the three OOD variants. **Best** and second-best per column, including ties.

Method	Cluster		Explosion		Implosion		Average	
	Gap↓	Det.↓	Gap↓	Det.↓	Gap↓	Det.↓	Gap↓	Det.↓
AM [41]	17.97%	678%	3.82%	65%	2.43%	5%	6.63%	249%
AMDKD+EAS [7]	0.17%	112%	0.05%	-39%	0.08%	1%	0.09%	25%
DIFUSCO [78]	2.87%	184%	1.38%	37%	2.80%	177%	2.02%	133%
T2T [48]	1.50%	733%	0.15%	-17%	2.60%	1344%	1.11%	687%
Fast-T2T [49]	1.18%	1867%	**0.03%**	**-52%**	2.50%	4067%	0.94%	1961%
GLOP [90]	0.17%	82%	0.07%	-28%	0.08%	**-10%**	0.10%	15%
EDISCO (this work)	**0.05%**	**25%**	**0.03%**	-25%	**0.05%**	13%	**0.04%**	**4%**

Table 3: CVRPs at $N \in \{50, 100, 200, 500\}$. Gap is relative to HGS-CVRP [80]. Time follows the protocol of Table 1. StruDiCO is adapted to CVRP from its TSP/MIS formulation. **Best** and second-best per Gap and Time among neural solvers. G: greedy, AE/NAE/NEE: adaptive / native adaptive / native edge expansion, LS: classical local-search post-processing. "–": not reported.

Method	Type	CVRP-50			CVRP-100			CVRP-200			CVRP-500		
		Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓
Reference Solvers													
HGS-CVRP [80]	Heuristic	10.37	0.00%	1.01 s	15.56	0.00%	20.03 s	19.63	0.00%	60.02 s	37.15	0.00%	240 s
Non-Generative Neural Solvers													
Sym-NCO [39]	RL+G	10.57	1.93%	0.09 s	15.93	2.38%	0.19 s	20.19	2.85%	0.36 s	–	–	–
COExpander [59]	SL+AE+LS	10.77	3.85%	**0.03 s**	16.22	4.25%	**0.06 s**	20.59	4.89%	**0.15 s**	39.12	5.34%	0.61 s
Generative Neural Solvers													
StruDiCO [84]	SL+G+LS	10.41	0.39%	0.5 s	15.61	0.32%	1.5 s	19.72	0.46%	4 s	–	–	–
NEXCO [85]	SL+NAE	10.48	1.12%	0.08 s	15.78	1.40%	0.12 s	20.11	2.45%	0.39 s	–	–	–
EDISCO (this work)	SL+NEE	10.40	**0.29%**	0.06 s	15.60	**0.26%**	0.10 s	19.71	**0.41%**	0.20 s	37.78	**1.70%**	**0.50 s**

Table 4: Constraint robustness on CVRP-100 across $Q \in \{10, 50, 100, 200, 300, 400, 500\}$ following Luo et al. [58]. Gap (%)↓ vs HGS-CVRP. **Best** and second-best per column among single-model solvers, with per-capacity specialists as an excluded-from-ranking upper bound. StruDiCO, NEXCO, and COExpander are trained at Q=50 and evaluated at all listed capacities without retraining.

Method	Q=10	Q=50	Q=100	Q=200	Q=300	Q=400	Q=500
HGS-CVRP [80]	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AM [41]	9.32	7.34	6.85	5.41	4.92	4.83	5.27
POMO [43]	4.18	3.09	2.95	2.74	2.81	3.05	3.42
Multi-expert [58]	1.05	0.61	0.74	0.85	0.92	1.01	1.12
CaR [8]	1.34	0.78	0.91	1.04	1.18	1.27	1.39
StruDiCO [84]	6.21	0.32	1.45	3.18	4.62	5.35	5.84
NEXCO [85]	5.84	1.40	1.62	3.05	4.31	4.92	5.18
COExpander [59]	6.95	4.25	2.38	3.94	4.85	5.42	5.71
EDISCO default-only	5.20	0.30	1.30	2.85	4.05	4.55	4.85
EDISCO mixed unconditioned	1.50	1.05	1.15	1.30	1.45	1.55	1.70
EDISCO capacity-conditioned (this work)	**0.55**	**0.26**	**0.35**	**0.45**	**0.55**	**0.65**	**0.78**
EDISCO per-capacity specialists (upper bound)	0.45	0.26	0.28	0.35	0.40	0.45	0.52

Capacity conditioning closes most of the gap between mixed unconditioned and per-capacity specialists within a single parameter set. Luo's multi-expert mixture and CaR remain in the 0.6–1.4% range, while the fixed-capacity baselines StruDiCO and NEXCO perform best at their training capacity and degrade rapidly away from it, and COExpander stays above 2% at every capacity.

4.4 Robustness to Training Data Variations

This subsection evaluates EDISCO's robustness to variations in training data quantity and quality, two factors that bear on practical deployment whenever obtaining optimal solutions is computationally expensive. The left panel of Figure 4 shows that EDISCO maintains near-optimal performance even with limited data, achieving gaps below 0.07% with just 10% of TSP-50 training data, while DIFUSCO reaches 2.8% and T2T 2.1% under the same setting.

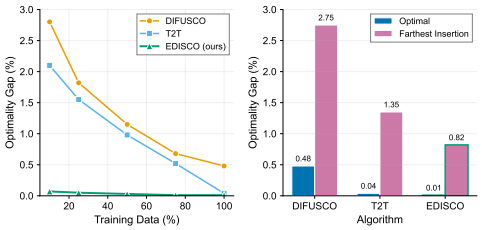

Figure 4: TSP-50 performance under training-data variations. Left: optimality gap as a function of training set size. Right: performance comparison when trained on optimal data versus heuristic Farthest Insertion data.

The right panel of Figure 4 examines model performance when trained on suboptimal solutions generated by the Farthest Insertion heuristic, which produces tours with an average gap of 7.5% to optimal on TSP-50 [48]. EDISCO achieves a 0.82% gap, outperforming DIFUSCO (2.75%) and T2T (1.35%). All methods use greedy decoding for this comparison.

4.5 Ablation Study

Table 5 isolates encoder and decoder choices on TSP-500/1000, comparing architectural orbit-sharing with finite-training orbit-sampling. Replacing the EGNN with a parameter-matched non-equivariant GNN nearly triples the gap. E(2) data augmentation [11] recovers about half of the lost performance in this ablation, while a Sym-NCO soft regularizer [39] falls further behind. NEE matches or exceeds greedy + 2-opt [17] on Gap at both scales, replacing external local search with an E(2)-invariant decoder. Against greedy + 2-opt, NEE lowers the TSP-500 gap from 0.18% to 0.018% at 0.28 s (vs. 0.19 s for greedy + 2-opt). At TSP-1000, both report 0.52% at the displayed precision, while NEE takes 1.28 s (vs. 0.37 s). These total per-instance inference times show the quality–runtime trade-off. Multi-step sampling and capacity conditioning are evaluated in Appendix F.2 and Section 4.3.

Edge-probability consistency under random E(2) transformations (Table 6) tests Corollary 2. The metric measures the maximum probability change under rotations, reflections, or translations, reaching zero for exact invariance. EDISCO reaches the numerical-precision floor, while augmentation reduces residual coordinate-frame dependence. The sample-efficiency curves (Figure 5) likewise show EDISCO below both non-equivariant variants in quality gaps at every training budget.

Table 5: Encoder and decoder ablations on TSP-500/1000 with the 5-step DEIS-2 solver. Encoder block fixes the decoder at greedy (raw-P_{ij} ranking with degree-2 and subtour feasibility). Decoder block fixes the encoder at full EDISCO, with NEE ranking by Equation 13. Bold marks the proposed configuration in each block.

Variant	Detail	TSP-500			TSP-1000		
		Length↓	Gap%↓	Time↓	Length↓	Gap%↓	Time↓
Encoder substitution (decoder fixed at greedy).							
EDISCO (Full)	Architectural E(2)	**16.90**	**2.10**	**0.11 s**	**23.83**	**3.05**	**0.31 s**
Non-equivariant GNN (param-matched)	None	17.53	5.95	0.12 s	24.93	7.85	0.32 s
+ E(2) data augmentation [11]	Augmentation	17.23	4.10	0.12 s	24.38	5.45	0.32 s
+ Sym-NCO soft equivariance loss [39]	Soft regularizer	17.35	4.85	0.12 s	24.63	6.55	0.32 s
Decoder choice (encoder fixed at full EDISCO).							
Greedy only	Native, invariant	16.90	2.10	0.11 s	23.83	3.05	0.31 s
Greedy + 2-opt [17]	External refinement	16.58	0.18	0.19 s	23.24	0.52	0.37 s
NEE (proposed)	Native, invariant	**16.55**	**0.018**	**0.28 s**	**23.24**	**0.52**	**1.28 s**

Table 6: Edge-probability consistency on TSP-100. Mean of $\max_{i,j} |P_{ij}(\mathbf{x}) - P_{ij}(g \cdot \mathbf{x})|$ over the 1,280 test instances of Kool et al. [41] and 16 random $g \in \mathrm{E}(2)$ per instance, with noisy edge states and diffusion times held fixed.

Encoder	Mean $\max_{i,j} \lvert\Delta P_{ij}\rvert \downarrow$
EDISCO	1.2×10^{-7}
Non-eq. + E(2) aug. [11]	1.7×10^{-2}
Non-eq. (no symmetry)	9.4×10^{-2}

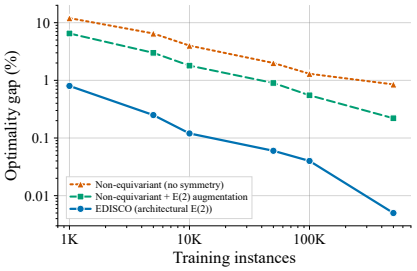

Figure 5: Sample efficiency on TSP-100 vs. training-set size.

5 Discussion and Conclusion

We presented EDISCO, a discrete diffusion solver for Euclidean combinatorial optimization with an E(2)-invariant distribution over node-index solutions. This invariance is preserved through edge-score prediction, categorical reverse sampling, sparse graph construction, and decoding into feasible solutions. Exact CTMC posterior sampling supports a flexible quality–speed trade-off without retraining. Across TSP and CVRP benchmarks, EDISCO achieves strong solution quality using fewer training instances and remains robust under spatial-distribution and capacity shifts.

The geometric guarantee applies to Euclidean problems such as the TSP, CVRP, and ESTP. For arbitrary-graph problems such as the MIS and Max-Cut, the optimum depends on topology rather than node coordinates, so the E(2)-specific contributions do not apply. The MIS experiment in Appendix A.3 evaluates the categorical CTMC engine with a non-equivariant GNN. Asymmetric routing and real-world non-Euclidean network costs also lie outside the present geometric guarantee.

These limitations motivate future extensions through directional encoders, as in MatNet [44] and GREAT [54], or coordinate representations of asymmetric costs learned through Finsler multi-dimensional scaling [18] and related metric-learning methods. Studying equivariance in these representations and closing the sim-to-real gap remain future work. Within the Euclidean setting, EDISCO combines an exact geometric guarantee with the solution quality, data efficiency, and robustness demonstrated in our experiments.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC; project numbers RES0048688, RES0051374, and RES0054326) and Alberta Innovates (project number RES0053965).

References

[1] 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.

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

[3] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993, 2021.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[6] 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*, 2025. arXiv:2406.15007.

[7] Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. Learning generalizable models for vehicle routing problems via knowledge distillation. In *Advances in Neural Information Processing Systems*, volume 35, pages 31226–31238, 2022.

[8] 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 *International Conference on Learning Representations*, 2026. arXiv:2602.16012.

[9] 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. URL `https://openreview.net/forum?id=mk0HzdqY7i1`.

[10] Marcus Brazil and Martin Zachariasen. *Optimal Interconnection Trees in the Plane: Theory, Algorithms and Applications*, volume 29 of *Algorithms and Combinatorics*. Springer, 2015. doi: 10.1007/978-3-319-13915-9. URL `https://doi.org/10.1007/978-3-319-13915-9`.

[11] Johann Brehmer, Sönke Behrends, Pim de Haan, and Taco Cohen. Does equivariance matter at scale? *Transactions on Machine Learning Research*, 2025. URL `https://openreview.net/forum?id=wilNute8Tn`.

[12] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[13] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[14] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, volume 35, pages 28266–28279, 2022.

[15] Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 32, pages 6278–6289, 2019.

[16] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999. PMLR, 2016.

[17] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6): 791–812, 1958. doi: 10.1287/opre.6.6.791.

[18] Thomas Dagès, Simon Weber, Ya-Wei Eileen Lin, Ronen Talmon, Daniel Cremers, Michael Lindenbaum, Alfred M. Bruckstein, and Ron Kimmel. Finsler multi-dimensional scaling: Manifold learning for asymmetric dimensionality reduction and embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 25842–25853, 2025.

[19] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

[20] 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. doi: 10.52202/075280-3385. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f445ba15f0f05c26e1d24f908ea78d60-Abstract-Conference.html.

[21] Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. GOAL: A generalist combinatorial optimization agent learner. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=z2z9suDRjw.

[22] Floor Eijkelboom, Grigory Bartosh, Christian A. Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *Advances in Neural Information Processing Systems*, volume 37, pages 11735–11764, 2024.

[23] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61, 1960.

[24] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical CNNs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[25] 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.

[26] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 15554–15566, 2019.

[27] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, pages 133345–133385, 2024.

[28] 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 *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 884–895, 2024.

[29] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2020. URL https://www.gurobi.com.

[30] Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via Min-SNR weighting strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7441–7451, 2023.

[31] Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017. URL http://www.akira.ruc.dk/~keld/research/LKH-3/LKH-3_REPORT.pdf.

[32] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[33] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8867–8887. PMLR, 2022. URL https://proceedings.mlr.press/v162/hoogeboom22a.html.

[34] 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. URL https://openreview.net/forum?id=ar92oEosBIg.

[35] 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.

[36] Chaitanya K. Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27(1–2): 70–98, 2022. doi: 10.1007/s10601-022-09327-y.

[37] Daniel Juhl, David M. Warme, Pawel Winter, and Martin Zachariasen. The GeoSteiner software package for computing Steiner trees in the plane: an updated computational study. *Mathematical Programming Computation*, 10(4):487–532, 2018. doi: 10.1007/s12532-018-0135-8. URL https://doi.org/10.1007/s12532-018-0135-8.

[38] Andrew B. Kahng and Gabriel Robins. A new class of iterative Steiner tree heuristics with good performance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(7):893–902, 1992.

[39] 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.

[40] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707, 2021.

[41] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

[42] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[43] 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.

[44] Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 34, pages 5138–5149, 2021.

[45] 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.

[46] John M. Lee. *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer, 2003. doi: 10.1007/978-0-387-21752-9.

[47] Han Li, Fei Liu, Zhi Zheng, Yu Zhang, and Zhenkun Wang. CaDA: Cross-problem routing solver with constraint-aware dual-attention. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 35438–35456. PMLR, 2025.

[48] 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.

[49] 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.

[50] 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 *Advances in Neural Information Processing Systems*, volume 38, pages 127168–127196, 2025. doi: 10.52202/085713-4237. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b8e9869fa827226ff68db9290659a20c-Abstract-Conference.html.

[51] 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, pages 537–546, 2018.

[52] Peijia Lin, Pin Chen, Rui Jiao, Qing Mo, Jianhuan Cen, Wenbing Huang, Yang Liu, Dan Huang, and Yutong Lu. Equivariant diffusion for crystal structure prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 29890–29913. PMLR, 2024. URL https://proceedings.mlr.press/v235/lin24b.html.

[53] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[54] Attila Lischka, Filip Rydin, Jiaming Wu, Morteza Haghir Chehreghani, and Balázs Kulcsár. A GREAT architecture for edge-based graph problems like TSP. *arXiv preprint arXiv:2408.16717*, 2025. URL https://arxiv.org/abs/2408.16717.

[55] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022.

[56] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, volume 35, pages 5775–5787, 2022.

[57] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *Advances in Neural Information Processing Systems*, volume 36, pages 8845–8864, 2023.

[58] Fu Luo, Yaoxin Wu, Zhi Zheng, and Zhenkun Wang. Rethinking neural combinatorial optimization for vehicle routing problems with different constraint tightness degrees. In *Advances in Neural Information Processing Systems*, volume 38, pages 127283–127317, 2025. arXiv:2505.24627.

[59] Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. COExpander: Adaptive solution expansion for combinatorial optimization. In *Forty-second International Conference on Machine Learning*, 2025.

[60] Pinaki Mohanty and Ruqi Zhang. Slithering through gaps: Capturing discrete isolated modes via logistic bridging. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pages 4600–4608. PMLR, 2026. URL https://proceedings.mlr.press/v300/mohanty26a.html.

[61] Pinaki Mohanty, Riddhiman Bhattacharya, and Ruqi Zhang. Entropy-guided sampling of flat modes in discrete spaces. In *NeurIPS 2025 Workshop on Structured Probabilistic Inference and Generative Modeling*, 2025. URL https://arxiv.org/abs/2505.02296.

[62] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[63] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 2021.

[64] Oskar Nordenfors, Fredrik Ohlsson, and Axel Flinth. Optimization dynamics of equivariant and augmented neural networks. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=PTTa3U29NR.

[65] James R. Norris. *Markov Chains*, volume 2 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, 1998. doi: 10.1017/CBO9780511810633.

[66] Wenbin Ouyang, Yisen Wang, Paul Weng, and Shaochen Han. Generalization in deep RL for TSP problems via equivariance and local search. *SN Computer Science*, 5(4):369, 2024.

[67] 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 *The Thirteenth International Conference on Learning Representations*, 2025.

[68] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[69] Jonathan Pirnay and Dominik G. Grimm. Self-improvement for neural combinatorial optimization: Sample without replacement, but improvement. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=agT8ojoH0X.

[70] Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 50269–50326. PMLR, 2025. URL https://proceedings.mlr.press/v267/qin25d.html.

[71] 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.

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

[73] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

[74] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10 of *Institute of Mathematical Statistics Textbooks*. Cambridge University Press, 2019. doi: 10.1017/9781108186735.

[75] Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9323–9332. PMLR, 2021.

[76] Ya Song, Laurens Bliek, and Yingqian Zhang. Geometrically invariant and equivariant graph neural networks for TSP algorithm selection and hardness prediction. In *Learning and Intelligent Optimization*, volume 15744 of *Lecture Notes in Computer Science*, pages 237–252. Springer, 2026. doi: 10.1007/978-3-032-09156-7_16.

[77] Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *International Conference on Learning Representations*, 2023.

[78] 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.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[80] Thibaut Vidal. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140:105643, 2022. arXiv:2012.10384.

[81] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700, 2015.

[82] Haoyu Peter Wang and Pan Li. Unsupervised learning for combinatorial optimization needs meta learning. In *International Conference on Learning Representations*, 2023.

[83] Siqi Wang, Yifan Wang, and Guangmo Tong. Deep-Steiner: Learning to solve the Euclidean Steiner tree problem. In *Wireless Internet*, volume 464 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 228–242. Springer, 2023. doi: 10.1007/978-3-031-27041-3_16. URL https://doi.org/10.1007/978-3-031-27041-3_16.

[84] 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 *Advances in Neural Information Processing Systems*, volume 38, pages 71201–71228, 2025. doi: 10.52202/085713-2394. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/6728fcf94660c59c938319a6833a6073-Abstract-Conference.html.

[85] Yu Wang, Yang Li, Jiale Ma, Junchi Yan, and Yi Chang. Native adaptive solution expansion for diffusion-based combinatorial optimization. In *International Conference on Learning Representations*, 2026. OpenReview 084SvT55yk.

[86] 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 *Advances in Neural Information Processing Systems*, volume 34, pages 7472–7483, 2021.

[87] Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artificial Intelligence*, 171(8-9): 514–534, 2007.

[88] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[89] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJxbJeHFPS.

[90] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20284–20292, 2024.

[91] Deunsol Yoon, Hyungseok Song, Kanghoon Lee, and Woohyung Lim. CADO: Cost-aware diffusion solvers for combinatorial optimization through RL fine-tuning. In *ICML 2024 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2024. URL https://www.lgresearch.ai/publication/view?seq=100.

[92] Dinghuai Zhang, Hanjun Dai, Esmeralda S. Whitammer, 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.

[93] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *International Conference on Learning Representations*, 2023.

[94] 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. doi: 10.1016/j.gmod.2025.101284.

[95] 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 *Advances in Neural Information Processing Systems*, volume 37, pages 6081–6125, 2024.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Section 1 state three contributions tied to Proposition 1 and the experiments in Section 4, with the Euclidean scope made explicit and non-Euclidean MIS bracketed out (Appendix A.3).

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 5 discusses the Euclidean scope, non-Euclidean and asymmetric-routing limitations, and sim-to-real transfer as future work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: Proposition 1 is stated with assumptions inline and proved in full in Appendix B.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Architecture, training configuration, data generation, and hardware/timing protocol are documented in Appendices D, E.6, E.4, and E.2; the public implementation is linked in the author footnote.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The link to the code is provided in the author footnote.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Optimizer, schedule, batch size, epochs, and architectural hyperparameters appear in Appendices E.5 and E.6, with test splits and protocol in Section 4.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: All EDISCO results are averaged over five inference runs with different random seeds (Appendix E.3); the $\pm 0.03\%$ run-to-run variance is well below the gap differences reported in Section 4.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix E.2 specifies the GPU and CPU configuration used for all timing measurements, and Appendix F.8 reports per-model training cost.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The paper introduces algorithmic methods for combinatorial optimization on synthetic and standard public benchmarks, with no human subjects, personal data, or sensitive content involved.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

Justification: EDISCO studies general-purpose combinatorial-optimization methods on academic benchmarks; foreseeable downstream uses follow established applications of TSP and CVRP solvers, with no direct societal-impact claim made.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The released artifacts are code for combinatorial optimization on algorithmic benchmarks, with no high-risk generative content or sensitive data.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All baseline implementations, datasets, and reference solvers are cited in Sections 2 and 4 and the bibliography, and their public licenses are respected.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The code repository contains the documentation needed.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not a component of the proposed method.

    Guidelines:

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

Appendices

Default Notation 25
A. Extensions to Other Combinatorial Problems 27
A.1 Euclidean Steiner Tree Problem 27
A.2 Symmetry-Faithful Partition Diffusion for Large-Scale CVRPs 28
A.3 Maximum Independent Set (Engine-Generality Evidence) 29
B. Proofs 29
B.1 Proof of Proposition 1 29
B.2 Proof of E(2)-Equivariance in EDISCO 31
C. Extended Related Work 34
D. Architecture Details 35
D.1 Network Architecture Overview 35
D.2 Feature Representations and Initialization 35
D.3 Equivariant Message Passing Mechanism 35
E. Additional Experiment Details 36
E.1 Performance Metrics 36
E.2 Hardware Platform 36
E.3 Randomness in Results 36
E.4 Data Generation Process 36
E.5 Model Architecture Specifications 36
E.6 Training Configuration 37
F. Additional Results 37
F.1 Multi-Step Methods for Categorical CTMC 37
F.2 Solver Evaluation on TSP-500 38
F.3 TSP-10000 Results 39
F.4 TSPLIB Results 39
F.5 Noise Schedule Design for TSP Diffusion 40
F.6 Evaluation on Architectural Hyperparameters 41
F.7 Model Efficiency 42
F.8 Training Resource Requirements 43
F.9 Visualization of EDISCO's E(2)-Equivariance 44
F.10 Compute and Runtime Conventions for Recent Competitors 45

Default Notation

Numbers and Arrays

a	A scalar (integer or real)
$\mathbf{v}$	A vector
$\mathbf{M}$	A matrix
$\mathcal{T}$	A tensor
$\mathbf{I}_n$	Identity matrix with n rows and n columns
$\mathbb{1}$	All-ones vector, with dimension determined by the expression (K in Equation 2)
$\mathbf{0}$	Vector or matrix of zeros (dimensionality implied by context)
$\text{diag}(\mathbf{v})$	A square, diagonal matrix with diagonal entries given by $\mathbf{v}$

Graph and Combinatorial Structures

$G = (V, E)$	A graph with vertex set V and edge set E
V	Set of n nodes/cities
$\mathbf{c}_i \in \mathbb{R}^2$	Coordinates of node/city i
$\mathbf{X} \in \{0,1\}^{n \times n}$	Binary adjacency matrix representing edges
X_{ij}	Element (i, j) of adjacency matrix (1 if edge exists, 0 otherwise)
d_{ij}	Euclidean distance between nodes i and j
$\mathcal{C}$	Constraint condition set for valid tours

Diffusion Process

$\mathbf{X}_0$	Clean edge-state matrix at time $t = 0$
$\mathbf{X}_t$	Edge-state matrix at diffusion time $t \in [0, 1]$, with $\mathbf{X}_0$ the clean state
$\beta(t)$	Time-dependent noise schedule
$\beta_{\min}, \beta_{\max}$	Minimum and maximum noise rates
$\mathbf{Q}(t)$	Rate matrix for continuous-time Markov chain
$\mathbf{P}(t\|s)$	Transition probability matrix from time s to t
K	Number of categorical states (2 for binary edges)
$q(\cdot)$	Forward diffusion distribution
$p_\theta(\cdot)$	Reverse diffusion distribution parameterized by θ

Neural Network Components

$\mathbf{h}_i^{(\ell)}$	Node features for node i at layer ℓ
$\mathbf{e}_{ij}^{(\ell)}$	Edge features between nodes i and j at layer ℓ
$\mathbf{x}_i^{(\ell)}$	Coordinates for node i at layer ℓ (evolve equivariantly)
$\mathbf{m}_{ij}^{(\ell)}$	Message from node j to node i at layer ℓ
s_θ	Score network with parameters θ
α	Step size for coordinate updates
τ	Temperature parameter for weight scaling

Geometric Transformations

$E(2)$	Euclidean group in 2D: $\mathbb{R}^2 \rtimes O(2)$ (translations, rotations, reflections)
$O(2)$	Orthogonal group (rotations and reflections)
$SO(2)$	Special orthogonal group (rotations only)
$g \in E(2)$	A Euclidean transformation
$g \cdot \mathbf{c}$	Action of transformation g on coordinates $\mathbf{c}$
X/G	Quotient space under group action
$\pi : X \to X/G$	Canonical projection to quotient space
$A \rtimes B$	Semi-direct product of groups A and B

Probability and Optimization

$p(\mathbf{X}\|\{\mathbf{c}_i\})$	Conditional distribution of tours given coordinates
$\mathbb{E}[\cdot]$	Expectation
$\mathrm{Cat}(\cdot)$	Categorical distribution
δ_{ij}	Kronecker delta (1 if $i = j$, 0 otherwise)
$\mathcal{L}$	Loss function
NFE	Number of score-network evaluations
$O(t^k)$	Big-O asymptotic order: $f(t) = O(t^k)$ means $\|f(t)\| \leq Ct^k$ for sufficiently small t
Gap	Optimality gap: $(L_{\text{pred}} - L_{\text{opt}})/L_{\text{opt}} \times 100\%$

Functions and Operations

$\|\cdot\|_2$	Euclidean norm
$\oplus$	Concatenation operation
$\odot$	Element-wise multiplication
$\circ$	Function composition, $(f \circ g)(x) = f(g(x))$
$\sigma(\cdot)$	Sigmoid activation function
$\tanh(\cdot)$	Hyperbolic tangent activation
$\mathrm{MLP}(\cdot)$	Multilayer perceptron
$\mathrm{LayerNorm}(\cdot)$	Layer normalization
$\mathrm{SiLU}(\cdot)$	Sigmoid linear unit activation
$\mathrm{softmax}(\cdot)$	Softmax function
$\mathrm{argmax}(\cdot)$	Argument of the maximum

A Extensions to Other Combinatorial Problems

This appendix collects EDISCO's extensions to three combinatorial problems. The first two, ESTP and CVRP, share Euclidean structure with TSP and inherit the E(2) symmetry argument of Appendix B.2. The third, MIS, is non-Euclidean and is included as engine-generality evidence for the categorical CTMC diffusion machinery, since the Euclidean symmetry contributions do not apply by construction.

A.1 Euclidean Steiner Tree Problem

Problem and equivariance preservation. The ESTP [10] asks for the minimum-length tree connecting a given set of terminal points in Euclidean space, with the option to add Steiner points that reduce total length. EDISCO targets the discrete candidate-graph variant in which Steiner-point locations are sampled in advance and the diffusion model selects which terminal-terminal, terminal-candidate, or candidate-candidate edges enter the tree. Unlike TSP, ESTP produces acyclic trees rather than cycles, providing a natural testbed for EDISCO beyond routing. The optimal Steiner tree under any rotation, translation, or reflection of the point set is the corresponding transformation of the original tree, since edge lengths are preserved by orthogonal transformations of the coordinates. EDISCO therefore reuses the E(2)-equivariant EGNN of Section 3 with terminal/Steiner identity as an invariant binary node feature,

$$\mathbf{h}_i^{(0)} = \text{NodeEmbed}([\mathbb{1}_{\text{terminal}}(i)]), \tag{14}$$

where $\mathbb{1}_{\text{terminal}}(i)$ marks whether node i is a required terminal or a candidate Steiner point. Coordinates remain the only equivariant input, and because the indicator is invariant under E(2), all scalar message passing remains E(2)-invariant by Corollary 2.

Tree-aware greedy decoding. Denoised edge probabilities are converted into a feasible Steiner tree by a Kruskal-style greedy decoder [42] on the symmetrized score $s_{ij} = (P_{ij} + P_{ji})/(d_{ij} + \epsilon)$. Starting from terminals $\mathcal{T}$ and candidate Steiner points $\mathcal{S}$, the highest-scoring edge that connects two distinct components is added at each step, with cycles rejected and Steiner points retained only when they reduce total length by serving as connection hubs. Any terminal still disconnected at the end is attached through its shortest remaining feasible edge, guaranteeing feasibility without breaking the equivariant computation. ESTP benchmarks with 10, 20, and 50 terminals follow Wang et al. [83] with an equal number of candidate Steiner points uniformly sampled from $[0, 1]^2$, and training reuses the TSP configuration of Appendix E.6.

Table 7: Results on Steiner-10/20/50. G: greedy. GeoSteiner is the exact reference, MST the upper bound from terminal-only spanning trees. All neural methods trained on the same 10,000-instance dataset.

Algorithm	Type	Steiner-10			Steiner-20			Steiner-50		
		Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓
GeoSteiner [37]	Exact	2.62	0.00%	24 s	3.63	0.00%	2.8 m	5.78	0.00%	18.5 m
MST (Upper Bound)	Heuristic	2.84	8.40%	<1 s	3.89	7.16%	<1 s	6.15	6.40%	<1 s
Steiner Insertion [38]	Heuristic	2.71	3.44%	<1 s	3.76	3.58%	<1 s	5.97	3.29%	1 s
Learning-Based Greedy Decoding										
Deep-Steiner [83]	RL+G	2.73	4.20%	2 s	3.81	4.96%	3 s	6.08	5.19%	7 s
DIFUSCO [78]	SL+G	2.77	5.73%	2 s	3.81	4.96%	3 s	6.05	4.67%	7 s
T2T [48]	SL+G	2.73	4.20%	2 s	3.76	3.58%	3 s	5.97	3.29%	8 s
Fast-T2T [49]	SL+G	2.68	2.29%	1 s	3.71	2.20%	2 s	5.91	2.25%	4 s
EDISCO with 50-step PNDM (this work)	SL+G	**2.66**	**1.53%**	2 s	**3.68**	**1.38%**	3 s	**5.87**	**1.56%**	8 s
EDISCO with 5-step DEIS-2 (this work)	SL+G	2.69	2.67%	**0.15 s**	3.72	2.48%	**0.35 s**	5.93	2.60%	**0.7 s**

Results. Table 7 compares EDISCO against classical and learning-based baselines at two quality–speed configurations. With 50-step PNDM and greedy decoding, EDISCO achieves consistent $\sim$1.5% gaps across all sizes (1.53%, 1.38%, 1.56% at Steiner-10/20/50), more than $2\times$ better than Steiner Insertion (3.44%, 3.58%, 3.29%) and 1.4–3.7$\times$ better than DIFUSCO (5.73%/4.96%/4.67%), T2T (4.20%/3.58%/3.29%), and Fast-T2T (2.29%/2.20%/2.25%) retrained on matched hardware. The same model also surpasses Deep-Steiner (4.20%, 4.96%, 5.19%) by 2.7–3.6$\times$. The 5-step DEIS-2 configuration drops inference to 0.15–0.7 s (9–13$\times$ faster than PNDM-50) while maintaining 2.48–2.67% gaps, still below the MST upper bound (8.40%, 7.16%, 6.40%) and the Deep-Steiner, DIFUSCO, and T2T baselines.

A.2 Symmetry-Faithful Partition Diffusion for Large-Scale CVRPs

The end-to-end EDISCO formulation of Section 3 treats CVRPs as a single edge-distribution problem, with customer-demand and capacity constraints. CVRP-500 remains within reach of this end-to-end formulation and is reported in main Table 3. At $N \in \{1000, 2000\}$, existing learning-based solvers adopt divide-and-conquer [90, 95] by splitting customers into clusters and solving each cluster as a sub-TSP. This subsection extends EDISCO to the same scales with a two-stage pipeline whose partition stage is itself a symmetry-faithful diffusion process. Stage 1 generates a customer-to-route assignment via categorical CTMC diffusion over a pairwise same-route indicator, and Stage 2 solves each cluster plus the depot as a small Euclidean TSP using the unmodified EDISCO TSP score network of Section 3. Both stages depend only on E(2)-invariant quantities, so the composed pipeline preserves distribution-level E(2)-invariance end-to-end.

Stage 1: partition diffusion and capacity projection. Let $\mathbf{A} \in \{0,1\}^{N\times N}$ be the symmetric pairwise same-route indicator with $A_{ij} = 1$ iff customers i and j share a route. The forward CTMC corrupts $\mathbf{A}$ toward the uniform Bernoulli distribution under the same linear $\beta(t)$ schedule used for TSP edge variables (Equation 2), and the reverse process samples $\mathbf{A}$ from a partition score network s_θ^{part} conditioned on the E(2)-invariant feature set $\{d_{ij}, d_i/Q, (d_i + d_j)/Q, |d_i - d_j|/Q, \lambda, \sum_i d_i/Q\}$, with $\lambda = Q/Q_{\text{default}}$ the capacity ratio and d_i the demand of customer i. HGS-CVRP [80] reference solutions provide supervision because each reference route induces a cluster, so A_{ij} is the corresponding same-route ground truth, and training reuses the SNR-weighted cross-entropy of Section 3. Since the sampled $\hat{\mathbf{A}}$ may violate capacity, a combinatorial projection extracts routes by spectral clustering on $\hat{\mathbf{A}}$ followed by capacity-feasible re-balancing, in which over-capacity clusters shed their lowest-affinity customers to the nearest feasible cluster until every cluster fits. The projection reads only invariant cluster affinities and demand sums, preserving E(2)-invariance by construction.

Stage 2 and end-to-end equivariance. Each route plus the depot is solved as a small Euclidean TSP (typically at most fifty nodes) by the unmodified EDISCO TSP score network with NEE, inheriting the invariance argument of Appendix B.2 directly. End-to-end equivariance follows from composition. The partition score network reads only pairwise distances and demand-derived scalars, the projection is purely combinatorial on $\hat{\mathbf{A}}$, and each sub-TSP inherits the invariant pipeline of Appendix B.2, so the full pipeline is E(2)-invariant in distribution. The proof structure mirrors Appendix B.2 after replacing the edge-probability output with the pairwise same-route output at the partition stage.

Diffusion vs. k-means partitioning. An ablation isolates the role of partition diffusion by replacing the diffusion partitioner with a coordinate k-means baseline while keeping the same projection and sub-TSP solver. The gap between the two pipelines measures the contribution of the diffusion partitioner directly.

Table 8: Large-scale CVRPs at $N \in \{1000, 2000\}$ via the partition-diffusion extension. LKH-3 [31] is the reference at these scales since HGS-CVRP becomes prohibitively slow beyond CVRP-500; end-to-end CVRP-50/100/200/500 results against HGS-CVRP appear in main Table 3. The k-means row substitutes Lloyd's algorithm on raw coordinates for the diffusion partitioner, with the projection and sub-TSP solver unchanged. Time is the per-instance average over the 100 test instances. The LKH-3 and UDC times are the test-set totals reported by Zheng et al. [95] divided by 100, and the remaining entries are measured at batch size 1, single-thread, on the hardware of Appendix E.2. **Best** and second-best per Gap and Time column; the k-means ablation is excluded from ranking.

Method	Type	CVRP-1000			CVRP-2000		
		Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓
Reference							
LKH-3 [31]	Heuristic	46.40	0.00%	8.4 s	64.90	0.00%	24 s
Divide-and-Conquer Comparators							
GLOP [90]	RL+Partition	46.30	−0.22%	2.4 s	64.80	−0.15%	5.8 s
UDC (x_250) [95]	RL+Partition	43.00	−7.33%	43 s	60.01	−7.53%	77 s
EDISCO Partition Diffusion							
EDISCO + k-means partitioning (ablation)	SL+kMeans+NEE	44.50	−4.09%	1.5 s	61.85	−4.70%	3.3 s
EDISCO + partition diffusion (proposed)	SL+PartDiff+NEE	42.92	**−7.50%**	**1.2 s**	59.90	**−7.70%**	**3.0 s**

The proposed pipeline achieves gaps of $-7.50\%/-7.70\%$ relative to LKH-3 at CVRP-1000/2000 in 1.2/3.0 s per instance, leading UDC's best configuration by $0.17/0.17$ percentage points on Gap while running over an order of magnitude faster (1.2 s against 43 s per instance, a $36\times$ speedup at CVRP-1000 and a $26\times$ speedup at CVRP-2000). Replacing the diffusion partitioner with k-means on raw coordinates (same projection and sub-TSP solver) reduces the improvement over LKH-3 from 7.50% to 4.09% at CVRP-1000 and from 7.70% to 4.70% at CVRP-2000, isolating the partition-stage contribution. Reported times in Table 8 cover the full pipeline (partitioning, capacity-feasible re-balancing, and per-cluster sub-TSP decoding), not partitioning alone. NEXCO and COExpander do not report at these scales.

A.3 Maximum Independent Set (Engine-Generality Evidence)

The MIS problem seeks the largest vertex subset of an undirected graph $G = (V, E)$ such that no two selected vertices share an edge. MIS is non-Euclidean, with no spatial coordinates and no E(2) action on its vertex set. The continuous-time categorical diffusion engine of Section 3 is problem-agnostic, and pairing it with a non-equivariant backbone gives an engine-only baseline outside the Euclidean problem class. The E(2)-equivariant EGNN is replaced by a 12-layer message-passing GNN at the same width as the TSP score network, the prediction target shifts from edge probabilities to node-inclusion probabilities $p(v_i \in S)$, and the CTMC formulation with PNDM/DEIS-2 solvers remains unchanged. Following Sun and Yang [78] and Li et al. [48, 49], evaluation uses RB graphs [87] (200–300 vertices) and Erdős-Rényi (ER) graphs [23] (700–800 vertices, $p = 0.15$), with KaMIS [45] reference solutions and the test split of Qiu et al. [71]. Greedy decoding considers vertices in descending order of predicted probabilities and adds a vertex only if it is not adjacent to any previously selected ones. Sampling draws 128 candidates and reports the best.

On both RB-[200-300] and ER-[700-800], EDISCO is the second-best learning-based decoder by optimality drop in both greedy and sampling regimes, behind GenSCO and ahead of StruDiCO, Fast-T2T, and COExpander. On runtime, EDISCO is the fastest or second-fastest learning-based decoder across both graph families and decoding settings in Table 9, at 0.08 s on RB graphs and $0.40/0.55$ s on ER graphs. Only single-step Fast-T2T on ER with greedy decoding (0.20 s) and GFlowNets on RB with sampling (0.06 s) are faster, both at a larger drop. The same continuous-time engine that drives the Euclidean results in Section 4 therefore generalizes to a non-Euclidean benchmark with only a backbone swap, supporting engine generality on combinatorial structure outside the Euclidean problem class.

B Proofs

B.1 Proof of Proposition 1

Proof. **Part (i):** The Euclidean group $\mathrm{E}(2) = \mathbb{R}^2 \rtimes \mathrm{O}(2)$ consists of translations ($\mathbb{R}^2$) and orthogonal transformations ($\mathrm{O}(2)$, which includes both rotations and reflections). As a Lie group, $\mathrm{E}(2)$ has dimension 3: two parameters for translation and one for rotation angle. Reflections form a discrete $\mathbb{Z}_2$ component that does not contribute to the continuous dimension. The action of $g = (t, R) \in \mathrm{E}(2)$ on a configuration $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in X$ is given by:

$$g \cdot (\mathbf{x}_1, \ldots, \mathbf{x}_n) = (R\mathbf{x}_1 + t, \ldots, R\mathbf{x}_n + t),$$

where $\mathbf{x}_i \in \mathbb{R}^2$ denotes the i-th city as a column vector and $R \in \mathrm{O}(2)$ acts by matrix multiplication.

Definition of general position. We define $X_{\mathrm{gp}} \subset X = (\mathbb{R}^2)^n$ as the set of configurations satisfying: (i) all n points are pairwise distinct, and (ii) no three points are collinear. For $n \geq 3$, this is an open dense subset of X with full measure (configurations violating these conditions form a measure-zero set).

The action is free on X_{gp}. We show that if $g \cdot \mathbf{x} = \mathbf{x}$ for some $g = (t, R) \in \mathrm{E}(2)$ and $\mathbf{x} \in X_{\mathrm{gp}}$, then g is the identity. The fixed-point condition gives, for every $i \in \{1, \ldots, n\}$:

$$R\mathbf{x}_i + t = \mathbf{x}_i.$$

For $i = 1$, $(R - I)\mathbf{x}_1 = -t$; for $i = 2$, $(R - I)\mathbf{x}_2 = -t$. Subtracting the second equation from the first gives

$$(R - I)(\mathbf{x}_1 - \mathbf{x}_2) = 0.$$

Table 9: Results on Maximum Independent Set. RB-[200-300]: random benchmark graphs with 200–300 vertices. ER-[700-800]: Erdős-Rényi graphs with 700–800 vertices, $p = 0.15$. G: greedy, S: sampling (128), TS: tree search, UL: unsupervised learning, AE: adaptive expansion, SC: search cycles, MIP: mixed-integer programming. Time is per instance. For KaMIS, Gurobi, Intel, DGL, LwD, DIMES, GFlowNets, DIFUSCO, T2T, and Fast-T2T it is the total test-set time reported by Li et al. [49] divided by the number of test graphs (500 for RB, 128 for ER). The remaining entries are measured at batch size 1, single-thread, on the hardware of Appendix E.2. †Gurobi is run under a time limit (50 minutes in total on the ER test set), and reported sizes are best feasible solutions found, not provable optima. **Best** and second-best per column among learning-based methods within each decoding block.

Algorithm	**Type**	**RB-[200-300]**			**ER-[700-800]**		
		Size↑	Drop↓	Time↓	Size↑	Drop↓	Time↓
KaMIS [45]	Heuristic	20.10	–	10.08s	44.87	–	24.44s
Gurobi [29]†	MIP	19.98	0.60%	5.71s	41.28	8.00%	23.44s
Learning-Based Methods with Greedy Decoding							
Intel [51]	SL+G	–	–	–	34.86	22.31%	2.84s
DIMES [71]	RL+G	–	–	–	38.24	14.78%	2.87s
DIFUSCO (T_s=100) [78]	SL+G	18.52	7.81%	1.93s	37.03	18.53%	2.58s
T2T (T_s=50, T_g=30) [48]	SL+G	18.98	5.49%	2.52s	39.81	11.28%	3.34s
Fast-T2T (T_s=1, T_g=1) [49]	SL+G	19.37	3.51%	0.16s	40.25	10.30%	**0.20s**
Fast-T2T (T_s=5, T_g=5) [49]	SL+G	19.49	2.89%	0.57s	40.68	9.34%	0.72s
StruDiCO (T_s=5, T_g=5) [84]	SL+G	19.75	1.71%	0.4s	42.13	6.32%	0.6s
GenSCO (C=500) [50]	SL+G+SC	**19.85**	**1.24%**	1.1m	**42.46**	**5.37%**	40s
COExpander (S=1, D_s=20) [59]	SL+AE	19.66	2.09%	0.1s	42.38	5.75%	0.5s
EDISCO (this work)	SL+G	19.83	1.34%	**0.08s**	42.40	5.51%	0.40s
Learning-Based Methods with Sampling/Search Decoding							
Intel [51]	SL+TS	18.47	8.11%	1.57s	38.80	13.43%	9.38s
DGL [9]	SL+TS	17.36	13.61%	1.53s	37.26	16.96%	10.65s
LwD [1]	RL+S	–	–	–	41.17	8.25%	2.97s
GFlowNets [92]	UL+S	19.18	4.57%	**0.06s**	41.14	8.53%	1.37s
DIFUSCO (T_s=100) [78]	SL+S	19.13	4.79%	2.46s	39.12	12.81%	10.18s
T2T (T_s=50, T_g=30) [48]	SL+S	19.38	3.53%	3.64s	41.41	7.72%	13.01s
Fast-T2T (T_s=1, T_g=1) [49]	SL+S	19.53	2.74%	0.24s	40.98	8.66%	0.62s
Fast-T2T (T_s=5, T_g=5) [49]	SL+S	19.70	1.90%	0.84s	41.73	6.99%	2.74s
StruDiCO (T_s=5, T_g=5) [84]	SL+S	19.91	0.93%	1.1s	43.27	3.79%	2.0s
GenSCO (C=8k) [50]	SL+SC	**20.04**	**0.29%**	17.6m	**44.13**	**1.66%**	10.7m
COExpander (S=4, D_s=20) [59]	SL+AE+S	19.71	1.88%	0.1s	42.56	5.34%	0.7s
EDISCO (this work)	SL+S	19.95	0.75%	0.08s	43.85	2.31%	**0.55s**

Let $\mathbf{v}_{12} = \mathbf{x}_1 - \mathbf{x}_2 \in \mathbb{R}^2$. Since $\mathbf{x} \in X_{\text{gp}}$, we have $\mathbf{v}_{12} \neq 0$ (distinct points). We consider two cases:

Case 1: $R \in \text{SO}(2)$ (rotation). Suppose $R \neq I$, so R is a rotation by angle $\theta \neq 0$. Then $(R - I)$ has eigenvalues $e^{i\theta} - 1$ and $e^{-i\theta} - 1$, both nonzero, so $(R - I)$ is invertible with $\ker(R - I) = \{0\}$. But $(R - I)\mathbf{v}_{12} = 0$ with $\mathbf{v}_{12} \neq 0$ implies $\mathbf{v}_{12} \in \ker(R - I)$, contradicting $\ker(R - I) = \{0\}$. Therefore $R = I$.

Case 2: $R \in \text{O}(2) \setminus \text{SO}(2)$ (reflection). A reflection has eigenvalues $+1$ and -1, so $(R - I)$ has rank 1 with $\ker(R - I)$ being the 1-dimensional reflection axis. The condition $(R - I)\mathbf{v}_{12} = 0$ forces $\mathbf{v}_{12}$ to lie along this axis. Similarly, for any other pair (i, j), the difference vector $\mathbf{v}_{ij}$ must also lie along the same axis. However, $\mathbf{x} \in X_{\text{gp}}$ implies no three points are collinear, so there exist difference vectors $\mathbf{v}_{12}$ and $\mathbf{v}_{13}$ that are linearly independent. These cannot both lie along a 1-dimensional axis, yielding a contradiction. Hence no non-identity reflection can fix $\mathbf{x}$.

In both cases, we conclude $R = I$, and then the fixed-point condition gives $t = 0$. Thus $g = e$ is the identity, establishing that the action is free on X_{gp}.

The action is proper. Properness means that the map $\Phi : \text{E}(2) \times X \to X \times X$, defined by $\Phi(g, \mathbf{x}) = (g \cdot \mathbf{x}, \mathbf{x})$, has compact preimages of compact sets [46]. Although $\text{E}(2)$ is non-compact, its action on $(\mathbb{R}^2)^n$ for $n \geq 2$ is proper. Let $K \subset X \times X$ be compact, and suppose $(g_k, \mathbf{x}_k)$ is a sequence in $\Phi^{-1}(K)$. Then $(g_k \cdot \mathbf{x}_k, \mathbf{x}_k) \in K$, so both $\mathbf{x}_k$ and $g_k \cdot \mathbf{x}_k$ lie in a bounded region.

Writing $g_k = (t_k, R_k)$, the translation t_k is determined by $t_k = g_k \cdot \mathbf{x}_1^{(k)} - R_k \mathbf{x}_1^{(k)}$, which remains bounded since both $\mathbf{x}_1^{(k)}$ and $g_k \cdot \mathbf{x}_1^{(k)}$ are bounded and $R_k \in \mathrm{O}(2)$ is compact. Since $\mathrm{O}(2)$ is compact, the sequence (R_k) has a convergent subsequence, and boundedness of (t_k) then implies (g_k) has a convergent subsequence in $\mathrm{E}(2)$. Thus $\Phi^{-1}(K)$ is compact, proving properness.

By the quotient manifold theorem [46], when a Lie group G acts freely and properly on a manifold M, the quotient M/G inherits a unique smooth manifold structure such that $\pi : M \to M/G$ is a smooth submersion. The dimension formula gives:

$$\dim(X_{\mathrm{gp}}/G) = \dim(X) - \dim(G) = 2n - 3.$$

Part (ii): Let $F : X_{\mathrm{gp}} \to Y$ be G-invariant (equivalently, G-equivariant with the trivial representation), meaning $F(g \cdot \mathbf{x}) = F(\mathbf{x})$ for all $g \in G$ and $\mathbf{x} \in X_{\mathrm{gp}}$.

For TSP edge prediction, the score network outputs continuous edge probabilities $Y = [0, 1]^{n \times n}$ (from which binary adjacency matrices are sampled); the representation is trivial, so the invariance condition holds.

Define $\widetilde{F} : X_{\mathrm{gp}}/G \to Y$ by $\widetilde{F}([\mathbf{x}]) = F(\mathbf{x})$, where $[\mathbf{x}] = \{g \cdot \mathbf{x} : g \in G\}$ denotes the orbit of $\mathbf{x}$. This is well-defined precisely because of invariance: if $[\mathbf{x}] = [\mathbf{x}']$, then $\mathbf{x}' = g \cdot \mathbf{x}$ for some $g \in G$, so:

$$\widetilde{F}([\mathbf{x}']) = F(\mathbf{x}') = F(g \cdot \mathbf{x}) = F(\mathbf{x}) = \widetilde{F}([\mathbf{x}]).$$

The factorization $F = \widetilde{F} \circ \pi$ follows immediately from the definition:

$$F(\mathbf{x}) = \widetilde{F}([\mathbf{x}]) = \widetilde{F}(\pi(\mathbf{x})) = (\widetilde{F} \circ \pi)(\mathbf{x}).$$

Uniqueness of $\widetilde{F}$ follows from surjectivity of π: if $F = \widetilde{F}_1 \circ \pi = \widetilde{F}_2 \circ \pi$, then for any $[\mathbf{x}] \in X_{\mathrm{gp}}/G$, choosing any representative $\mathbf{x} \in [\mathbf{x}]$:

$$\widetilde{F}_1([\mathbf{x}]) = \widetilde{F}_1(\pi(\mathbf{x})) = F(\mathbf{x}) = \widetilde{F}_2(\pi(\mathbf{x})) = \widetilde{F}_2([\mathbf{x}]).$$

Part (iii): The factorization $F = \widetilde{F} \circ \pi$ establishes a bijection between:

$$\{G\text{-invariant functions } X_{\mathrm{gp}} \to Y\} \longleftrightarrow \{\text{functions } X_{\mathrm{gp}}/G \to Y\}$$
$$F \longmapsto \widetilde{F}$$
$$\widetilde{F} \circ \pi \longleftarrow \widetilde{F}.$$

Therefore, learning any G-invariant function F is equivalent to learning the corresponding function $\widetilde{F}$ on the quotient manifold. Since X_{gp}/G has dimension $2n - 3$ while X has dimension $2n$, the domain of $\widetilde{F}$ has three fewer degrees of freedom.

In terms of function approximation, this means:

- Local function representations on X_{gp}/G use $2n - 3$ coordinates.
- Local charts for X_{gp}/G have dimension $2n - 3$.
- Equivalent coordinate frames share a single representation on the quotient.

This completes the proof that E(2)-invariant learning reduces to learning on a lower-dimensional manifold, providing a geometric motivation for improved data efficiency. □

B.2 Proof of E(2)-Equivariance in EDISCO

Proof. We prove that E(2)-equivariance is preserved throughout the entire EDISCO pipeline, from input processing through diffusion to tour construction.

Step 1: EGNN Architecture Preserves Equivariance

For any Euclidean transformation $g \in \mathrm{E}(2)$, we show each layer maintains equivariance:

(i) Distance invariance: For transformed coordinates $g \cdot c_i$ where $g = (t, R)$:

$$d_{ij}^{(g)} = \|g \cdot c_i - g \cdot c_j\|_2 = \|(Rc_i + t) - (Rc_j + t)\|_2 = \|R(c_i - c_j)\|_2 = \|c_i - c_j\|_2 = d_{ij},$$

where the last equality uses orthogonality of $R \in \mathrm{O}(2)$.

(ii) Message invariance: From Equation 7, messages depend only on:

- Node features h_i, h_j: Initial node features are built from E(2)-invariant scalar inputs. For TSPs, $\mathbf{h}_i^{(0)}$ is a single shared learnable embedding vector with no dependence on node identity or position. For ESTPs (Appendix A.1), $\mathbf{h}_i^{(0)}$ is computed from the binary terminal indicator $\mathbb{1}_{\text{terminal}}(i)$. For CVRPs (Section 3.3), $\mathbf{h}_i^{(0)}$ is augmented with the demand-derived scalar d_i/Q. All such inputs are invariant under any $g \in \mathrm{E}(2)$ acting on coordinates.
- Edge features e_{ij}: Initialized from the noisy adjacency matrix X_t and, for CVRPs, the invariant scalars $(d_i + d_j)/Q$ and $|d_i - d_j|/Q$, all of which are invariant.
- Layer-ℓ pairwise distances $\|\mathbf{x}_i^{(\ell)} - \mathbf{x}_j^{(\ell)}\|_2$: invariant by induction on ℓ. The base case ($\ell = 0$) follows from distance invariance of the input coordinates proven above. For the inductive step, suppose layer-ℓ coordinates transform as $\mathbf{x}_i^{(\ell)} \to R\mathbf{x}_i^{(\ell)} + t$ under $g \in \mathrm{E}(2)$ (verified by Step (iii) below). Then $\|R\mathbf{x}_j^{(\ell)} - R\mathbf{x}_i^{(\ell)}\|_2 = \|R(\mathbf{x}_j^{(\ell)} - \mathbf{x}_i^{(\ell)})\|_2 = \|\mathbf{x}_j^{(\ell)} - \mathbf{x}_i^{(\ell)}\|_2$ by orthogonality of R, so layer-ℓ pairwise distances remain invariant for all ℓ.

Therefore: $m_{ij}^{(g)} = m_{ij}$.

(iii) Coordinate equivariance: Consider the update rule (Equation 8). Writing $g = (t, R)$ where $t \in \mathbb{R}^2$ is the translation and $R \in \mathrm{O}(2)$ is the rotation/reflection, recall that g acts on coordinates as $g \cdot x_i = Rx_i + t$. For displacement vectors, translations cancel: $g \cdot x_j - g \cdot x_i = R(x_j - x_i)$. Thus:

$$\Delta(g \cdot x_i) = \alpha \sum_{j \neq i} w_{ij} \cdot \frac{g \cdot x_j - g \cdot x_i}{\|g \cdot x_j - g \cdot x_i\|_2 + \epsilon} \tag{15}$$

$$= \alpha \sum_{j \neq i} w_{ij} \cdot \frac{R(x_j - x_i)}{\|R(x_j - x_i)\|_2 + \epsilon} \tag{16}$$

$$= \alpha \sum_{j \neq i} w_{ij} \cdot \frac{R(x_j - x_i)}{\|x_j - x_i\|_2 + \epsilon} \quad \text{(orthogonality of } R\text{)} \tag{17}$$

$$= R \left(\alpha \sum_{j \neq i} w_{ij} \cdot \frac{x_j - x_i}{\|x_j - x_i\|_2 + \epsilon} \right) = R \cdot \Delta x_i. \tag{18}$$

Since coordinate updates transform as $\Delta(g \cdot x_i) = R \cdot \Delta x_i$, we have $(g \cdot x_i)^{(\ell+1)} = g \cdot x_i + R \cdot \Delta x_i = R(x_i + \Delta x_i) + t = g \cdot x_i^{(\ell+1)}$, confirming equivariance.

(iv) Feature invariance: Edge and node feature updates depend only on invariant quantities, thus $e_{ij}^{(g)} = e_{ij}$ and $h_i^{(g)} = h_i$.

Proof of Corollary 2. Step 1(i)–(iv) above establishes, by induction on layer ℓ, that pairwise distances and the scalar features $\mathbf{m}_{ij}^{(\ell)}$, $\mathbf{e}_{ij}^{(\ell)}$, $\mathbf{h}_i^{(\ell)}$ are E(2)-invariant, while the coordinate features $\mathbf{x}_i^{(\ell)}$ transform equivariantly, establishing the identities in Corollary 2. Each E(2)-invariant scalar feature map is constant on every E(2)-orbit by definition of invariance, so the same learned scalar feature map applies uniformly across the orbit of any input configuration. On configurations in general position, Proposition 1(ii) further identifies this orbit-constancy with unique factorization through the smooth quotient X_{gp}/G. □

Step 2: k-Nearest-Neighbor Sparsification Preserves Equivariance

For instances at $N \in \{500, 1000, 10000\}$, EDISCO restricts message passing to a k-nearest-neighbor graph constructed from pairwise distances (Section 4), with deterministic tie-breaking by node indices.

Let $\mathcal{N}_k(i)$ denote the k-nearest neighbors of node i. Since pairwise distances d_{ij} are E(2)-invariant by Step 1(i), the ordering of neighbors by distance is preserved under any $g \in \mathrm{E}(2)$:

$$d_{ij} < d_{ik} \iff d_{ij}^{(g)} < d_{ik}^{(g)}.$$

Exact distance ties are resolved by lower node index, a coordinate-independent rule. Therefore $\mathcal{N}_k^{(g)}(i) = \mathcal{N}_k(i)$, the message-passing graph is identical under transformation, and the invariance argument of Step 1 carries over from the dense graph to the sparsified graph without modification.

Step 3: Capacity Conditioning Preserves Equivariance

For CVRPs, capacity information enters the score network only through E(2)-invariant scalar channels (Section 3.3). The capacity embedding $\mathbf{z}_c = \mathrm{MLP}([\log \lambda, \sum_i d_i/Q])$ is built from the capacity ratio $\lambda = Q/Q_{\text{default}}$ and the demand-to-capacity sum, both invariant under coordinate transformations. The FiLM modulation in Equation 11,

$$\tilde{\mathbf{m}}_{ij}^{(\ell)} = \boldsymbol{\gamma}_\ell(\mathbf{z}_c) \odot \mathbf{m}_{ij}^{(\ell)} + \boldsymbol{\beta}_\ell(\mathbf{z}_c),$$

acts only on the scalar message channels. Since $\mathbf{m}_{ij}^{(\ell)}$ is invariant by Step 1(ii) and $\boldsymbol{\gamma}_\ell(\mathbf{z}_c), \boldsymbol{\beta}_\ell(\mathbf{z}_c)$ are coordinate-independent, the modulated message $\tilde{\mathbf{m}}_{ij}^{(\ell)}$ is also invariant. The coordinate update in Equation 8 is unchanged, so the only directional term remains $(\mathbf{x}_j - \mathbf{x}_i)$. For every fixed capacity value Q, the conditional distribution $p_\theta(\mathbf{X} \mid \{\mathbf{c}_i\}, Q)$ is therefore E(2)-invariant on coordinates.

Step 4: Diffusion Process Maintains Equivariance

The categorical diffusion operates on edge variables $X_{ij} \in \{0, 1\}$, which represent whether edge (i, j) is in the tour. These are inherently invariant to coordinate transformations.

Forward process: The corruption adds noise to edge selections independent of coordinates:

$$q(\mathbf{X}_t|\mathbf{X}_0) = \prod_{i,j} P_{X_{0,ij},\, X_{t,ij}}(t \mid 0),$$

where (i, j) ranges over node pairs and the transition factor $P_{X_{0,ij},\, X_{t,ij}}(t \mid 0)$ instantiates Equation 3 with categorical-state indices set to the initial and noisy edge values.

Reverse process: Since the score network s_θ outputs edge probabilities that are invariant (Steps 1–3), the reverse process maintains this invariance:

$$p_\theta(X_{t-\Delta t}|X_t, g(\{c_i\})) = p_\theta(X_{t-\Delta t}|X_t, \{c_i\}).$$

Step 5: Native Edge Expansion Preserves Equivariance

Native Edge Expansion (Section 3) converts an invariant edge probability matrix $P \in [0, 1]^{n\times n}$ into a feasible tour through an iterative procedure that reads only invariant inputs. For TSP, candidate edges are ranked by

$$r_{ij}^{(g)} = \frac{P_{ij}^{(g)} + P_{ji}^{(g)}}{2(d_{ij}^{(g)} + \epsilon)} = \frac{P_{ij} + P_{ji}}{2(d_{ij} + \epsilon)} = r_{ij},$$

since P_{ij} is invariant by Step 4 and d_{ij} is invariant by Step 1(i). Each feasibility check performed by the decoder depends only on E(2)-invariant quantities.

- Node degrees in the partial edge set S_t depend only on S_t, which carries no coordinate information.
- Subtour status, tracked by union-find on node indices, depends only on S_t.
- Hamiltonian closure tests cycle length against n, a coordinate-independent quantity.
- For CVRPs, accumulated route demand $\sum_{i\in\text{route}} d_i$ and the capacity bound Q are scalar and invariant.

Ties in r_{ij} are broken lexicographically by node indices, a coordinate-independent rule. The sequence of accepted edges is therefore determined entirely by invariant ranks and invariant feasibility tests, so for any $g \in \mathrm{E}(2)$ the procedure returns the identical edge set in node-index space:

$$\mathrm{NEE}(\hat{\mathbf{X}}_0, \{g \cdot \mathbf{c}_i\}) = \mathrm{NEE}(\hat{\mathbf{X}}_0, \{\mathbf{c}_i\}).$$

Composing Steps 1–5 establishes that the full EDISCO score-to-tour pipeline is E(2)-invariant in distribution. □

C Extended Related Work

Foundational neural combinatorial optimization. Pointer Networks [81] introduced sequence-to-sequence learning for combinatorial structures, initially requiring supervised labels from exact solvers. RL approaches later removed the need for labeled optimal tours [5]. AM [41] and POMO [43] established strong autoregressive baselines for TSPs and CVRPs. These models demonstrated that neural policies can learn useful construction heuristics, but generalization to larger or shifted instances remains challenging [25, 36].

Scaling beyond standard routing benchmarks. Large-scale neural routing has motivated architectures that reduce the cost of encoding, decoding, or search. LEHD [57] separates lightweight encoding from heavier decoding to improve large-scale generalization. BQ-NCO [20] uses bisimulation quotienting for stronger zero-shot generalization. GLOP [90] and UDC [95] use divide-and-conquer strategies for large CVRPs, while hierarchical real-world TSP solvers operate across multiple spatial resolutions [28]. UniCO [67] and GOAL [21] pursue broader problem unification. These works show that practical neural CO increasingly depends on scale, robustness, and transfer rather than only synthetic benchmark gaps.

Discrete diffusion and flow-based generative models. Discrete diffusion models provide a natural substrate for categorical structures [3]. Related generative modeling frameworks include continuous diffusion for categorical data [19], variational flow matching [22], discrete flow matching [27], and optimal-transport-inspired flow matching [53]. In graph generation, DeFoG [70] demonstrates the strength of discrete flow matching. These developments are complementary to EDISCO: they improve generative modeling machinery, while EDISCO studies exact Euclidean symmetry for edge-valued combinatorial diffusion.

Gradient-based discrete sampling. EDLP [61] uses a local-entropy auxiliary variable to guide discrete Langevin proposals toward flat modes, while HiSS [60] uses logistic noising to propose moves across barriers. These methods improve exploration of an instance-specific energy landscape over candidate solutions, targeting flat modes or transitions across barriers. EDISCO's geometric inductive bias concerns the two-dimensional Euclidean node layout of TSP and CVRP instances. It learns recurring layout patterns from solution labels and represents their relative geometry consistently under rotations, reflections, and translations. EDLP and HiSS explore each new instance's energy landscape through Markov chain Monte Carlo (MCMC), whereas EDISCO reuses one trained conditional model to predict edge decisions in a few denoising steps. EDISCO's TSP evaluations span 100 to 10000 nodes.

Theory, alignment, and data efficiency. The expressiveness of GNNs is limited by their aggregation structure [88], and algorithmic alignment argues that architectures generalize better when their computations match the target algorithmic structure [89]. For Euclidean CO, equivariance supplies such alignment by quotienting out coordinate-frame transformations. An empirical scaling study finds improved data and compute efficiency from equivariance, while augmentation can close the data-efficiency gap given sufficient training [11]. Work on TSP generalization also identifies limits of purely supervised learning [36]. These results motivate EDISCO's use of exact distribution-level E(2) invariance rather than augmentation-only symmetry learning.

Hybrid neural-classical optimization. Many practical solvers combine learned predictions with classical optimization. Neural local rewriting learns to improve existing solutions through local modifications [15]; learned branching and neural diving accelerate mixed-integer programming [26, 62]; and NeuroLKH [86] uses neural edge scores to guide LKH search. Other lines reduce label dependence through unsupervised learning or self-improvement [82, 69]. EDISCO follows the same practical principle that learned structure and combinatorial feasibility should be composed carefully, but its main focus is preserving Euclidean symmetry from the diffusion score model through sampling and decoding.

D Architecture Details

D.1 Network Architecture Overview

The EDISCO model employs a 12-layer EGNN that processes city coordinates and noisy adjacency matrices while maintaining geometric equivariance. The architecture consists of three main components: an embedding module, stacked equivariant layers, and a prediction head.

D.2 Feature Representations and Initialization

The model maintains three distinct feature types throughout the network:

Node Features. Node embeddings $\mathbf{h} \in \mathbb{R}^{n \times 64}$ are initialized from a single shared learnable embedding vector repeated across nodes, ensuring E(2)-invariance from the start. City coordinates $\mathbf{x} \in \mathbb{R}^{n \times 2}$ are maintained separately and used only to compute pairwise distances (which are rotationally invariant) in the message passing mechanism. Coordinates evolve through equivariant updates during message passing but never directly enter node feature computations.

Relational Features. Edge features $\mathbf{e} \in \mathbb{R}^{n \times n \times 64}$ encode pairwise relationships and tour decisions. These are initialized from the noisy adjacency matrix $\mathbf{X}_t$ through a single linear transformation.

Temporal Encoding. The continuous diffusion time $t \in [0, 1]$ is encoded using sinusoidal basis functions with frequencies spanning multiple octaves, producing a 128-dimensional representation that modulates the network's behavior at different noise levels.

D.3 Equivariant Message Passing Mechanism

The message, coordinate, edge, and node updates of each EGNN layer are given by Equations 7–10 of Section 3. This subsection lists the MLP component specifications required for reproducibility. With node dimension $d_n = 64$, edge dimension $d_e = 64$, and hidden dimension $d_h = 256$:

- **Message MLP (f_{msg})**: 3-layer MLP with architecture $[2d_n + d_e + 1 \to d_h \to d_h \to d_h]$, i.e., $[193 \to 256 \to 256 \to 256]$. Uses SiLU activation and layer normalization after the first linear layer, SiLU after the second linear layer. Input is the concatenation of $\mathbf{h}_i^{(\ell)}$, $\mathbf{h}_j^{(\ell)}$, $\mathbf{e}_{ij}^{(\ell)}$, and the layer-ℓ pairwise distance $\|\mathbf{x}_i^{(\ell)} - \mathbf{x}_j^{(\ell)}\|_2$.
- **Node Update MLP (f_{node})**: 2-layer MLP with architecture $[d_n + d_h \to d_h \to d_n]$, i.e., $[320 \to 256 \to 64]$. Uses SiLU activation and layer normalization after the hidden layer. Input is the concatenation of the current node feature $\mathbf{h}_i$ (64-dim) and aggregated messages $\sum_{j \neq i} \mathbf{m}_{ij}$ (256-dim).
- **Edge Update MLP (f_{edge})**: 2-layer MLP with architecture $[d_e + d_h \to d_h \to d_e]$, i.e., $[320 \to 256 \to 64]$. Uses SiLU activation and layer normalization after the hidden layer. Input is the concatenation of current edge feature $\mathbf{e}_{ij}$ (64-dim) and message $\mathbf{m}_{ij}$ (256-dim).
- **Time Embedding**: Two-stage processing. First, the sinusoidal time encoding (128-dim) passes through a shared MLP with architecture $[128 \to 256 \to 128]$ using SiLU activations. Then, per-layer time injection MLPs with architecture $[128 \to 64]$ (with SiLU) modulate edge features multiplicatively as $\mathbf{e}_{ij} \leftarrow \mathbf{e}_{ij} \cdot (1 + f_{\text{time}}(\mathbf{t}))$.
- **Coordinate Weight MLP (MLP_c)**: 2-layer MLP with architecture $[d_h \to d_h \to 1]$, i.e., $[256 \to 256 \to 1]$. Uses SiLU activation after the hidden layer, followed by temperature-scaled $\tanh(\cdot/\tau)$ with $\tau = 10$ to compute $w_{ij} = \tanh(\text{MLP}_c(\mathbf{m}_{ij})/\tau)$. The final layer has no bias to keep the coordinate-update gain centered at initialization (E(2)-invariance is already established by the invariant message input, independent of this bias choice). Input is the message $\mathbf{m}_{ij}$ (256-dim).

All MLPs use PyTorch default initialization (Kaiming uniform for weights). Layer normalization [4] is applied with learnable affine parameters (γ, β) initialized to 1 and 0 respectively. Residual connections with layer normalization are applied after node and edge updates.

E Additional Experiment Details

E.1 Performance Metrics

We evaluate models using three criteria:

- **Tour Length**: Average Euclidean length of generated tours across test instances
- **Optimality Gap**: Relative deviation from optimal/best-known solutions, computed as $(L_{\text{pred}} - L_{\text{opt}})/L_{\text{opt}} \times 100\%$
- **Inference Duration**: Per-instance wall-clock time, measured at batch size 1 in single-thread mode following the protocol of Wang et al. [84, 85], in seconds (s) or minutes (m)

Sampling-Based Decoding For SL+S+2O results (Sampling with 2-opt), we use a sampling budget of 128 samples per instance, following the protocol from Sun and Yang [78], Li et al. [48]. For each instance, 128 independent tours are generated using the diffusion model, 2-opt local search is applied to each tour, and the best tour (minimum length) is selected as the final solution. For TSPLIB experiments, we use $4\times$ sampling (4 samples per instance) following Li et al. [48] for fair comparison. Per-instance Time entries include the full sampling and 2-opt processing.

E.2 Hardware Platform

All experiments were conducted on a single NVIDIA RTX A6000 GPU paired with dual Intel Xeon Gold 5218R CPUs. Both training and inference use the same hardware configuration. For fair timing comparison, every reported neural-baseline timing is re-measured on this hardware at batch size 1 in single-thread mode, following the convention of Wang et al. [84, 85]. The exceptions are the baseline times in Tables 8 and 9 that their captions identify as per-instance averages of the test-set totals reported in the source papers. Classical solvers (Concorde, LKH-3, HGS-CVRP) are similarly run on the same hardware under their default trial counts, and per-instance Time is the average over the test set.

E.3 Randomness in Results

Due to the stochastic nature of diffusion models, all reported EDISCO results are averaged over five inference runs with different random seeds, and the reported run-to-run variation in optimality gap is within ± 0.03 percentage points. Baselines evaluated with fixed seeds are reported as single runs.

E.4 Data Generation Process

Instance Creation We follow the exact data generation protocol from DIFUSCO [78] for fair comparison. All cities are sampled uniformly from the unit square $[0, 1]^2$ following standard practice in the TSP literature. For smaller instances, TSP-50 and TSP-100 problems are solved to optimality using the Concorde exact solver [2] to obtain ground truth tours. For larger scales, TSP-500 and TSP-1000 instances are labeled using the LKH-3 heuristic solver [31] with 500 trials to ensure near-optimal solution quality. Our evaluation employs the standard test sets from Kool et al. [41] for TSP-50/100 containing 1,280 instances each, and from Fu et al. [25] for TSP-500/1000 containing 128 instances each.

Graph Sparsification. The kNN sparsification of Section 4 uses $k = 50$ for TSP-500 and $k = 100$ for TSP-1000 and TSP-10000.

E.5 Model Architecture Specifications

Unless stated otherwise, the full EDISCO network contains approximately 5.5M trainable parameters distributed across:

- 12 EGNN layers with shared architecture
- Node dimension: 64
- Edge dimension: 64

- Hidden dimension: 256
- Timestep embedding dimension: 128

This setting ensures that EDISCO has a similar number of trainable parameters to the state-of-the-art diffusion TSP solvers (5.3M) [78, 48, 91, 94], allowing for a fair comparison.

E.6 Training Configuration

We train EDISCO using the AdamW optimizer with a learning rate of 2×10^{-4} and weight decay of 10^{-5}. The learning rate follows a cosine annealing schedule over the training epochs to ensure smooth convergence. For training stability, we apply gradient clipping at unit norm to prevent exploding gradients during score-network training. The training objective minimizes weighted cross-entropy:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\mathbf{X}_0\sim p_{\text{data}},\mathbf{X}_t\sim q(\mathbf{X}_t|\mathbf{X}_0)} \left[(1-\sqrt{t}) \cdot \text{CE}(s_\theta(\mathbf{X}_t, t), \mathbf{X}_0) \right]. \tag{19}$$

The weighting $(1-\sqrt{t})$ is motivated by SNR considerations in diffusion training [40, 30]. As shown by Kingma et al. [40], the variational bound in continuous-time diffusion depends on SNR, and loss weighting that emphasizes cleaner timesteps (low t) improves optimization. The Min-SNR weighting strategy [30] demonstrates that balancing loss contributions across timesteps accelerates convergence by 3.4×. The weighting $(1-\sqrt{t})$ adopted here follows the same principle: it assigns a higher weight near $t = 0$ where reconstruction targets are reliable and lower weight near $t = 1$ where inputs are dominated by noise, with the square-root term controlling how weights decay across timesteps. The corresponding per-step training loop is given as Algorithm 1 in Section 3.

- **TSP-50**: 500,000 training instances, batch size 64, 100 epochs.
- **TSP-100**: 500,000 training instances, batch size 32, 100 epochs.
- **TSP-500**: 60,000 instances, batch size 16, 50 epochs with curriculum learning initialized from the TSP-100 checkpoint with best validation performance.
- **TSP-1000**: 30,000 instances, batch size 8, 50 epochs with curriculum learning initialized from the TSP-100 checkpoint with best validation performance.
- **TSP-10000**: 3,000 instances, batch size 4, 50 epochs with curriculum learning initialized from the TSP-500 checkpoint with best validation performance.
- **CVRP-50**: 500,000 HGS-CVRP-labeled training instances, batch size 64, 50 epochs.
- **CVRP-100**: 250,000 HGS-CVRP-labeled training instances, batch size 32, 50 epochs.
- **CVRP-200**: 16,000 HGS-CVRP-labeled training instances, batch size 16, 50 epochs with curriculum learning initialized from the CVRP-100 checkpoint with best validation performance.
- **CVRP-500**: 6,000 HGS-CVRP-labeled training instances, batch size 8, 50 epochs with curriculum learning initialized from the CVRP-200 checkpoint with best validation performance.

The CVRP training counts above are 39%/39%/50%/47% of the 1.28M/640K/32K/12.8K reported by COExpander (Appendix H.2 of Ma et al. [59]) at the matching scales. NEXCO reports the same training counts for CVRP-50/100/200 (Section 4.3 of Wang et al. [85]). These ratios fall within the same 33–50% sample-efficiency band that EDISCO achieves on TSPs.

F Additional Results

F.1 Multi-Step Methods for Categorical CTMC

Section 3.4 gives the per-edge categorical reverse kernel. Here we detail the multi-step prediction smoother and the posterior marginalization underlying that kernel.

Prediction smoothing via Adams-Bashforth-style coefficients Multi-step methods stabilize $\mathbf{X}_0$ prediction by linearly combining recent clean-state estimates. We adapt the multi-step structure of PNDM [55] to discrete CTMCs by applying the 4th-order Adams-Bashforth coefficients as an empirical state smoother: given predictions $\hat{\mathbf{X}}_0^{(t_1)}, \ldots, \hat{\mathbf{X}}_0^{(t_4)}$ at recent timesteps, ordered from newest to oldest,

$$\tilde{p}(\mathbf{X}_0) = \frac{55\hat{p}^{(t_1)} - 59\hat{p}^{(t_2)} + 37\hat{p}^{(t_3)} - 9\hat{p}^{(t_4)}}{24}. \tag{20}$$

The coefficients are originally derived for derivative integration in explicit ordinary differential equation (ODE) solvers; we use them as an affine combination of recent state predictions (the coefficients still sum to one) to stabilize predictions through temporal coherence. Because the combination is not a convex combination, the smoothed $\tilde{p}$ can fall outside $[0, 1]$, and we clamp it to $[0, 1]$ before passing $\tilde{p}$ to the posterior in Equation 12.

Posterior marginalization. Conditioning on x_0, the Markov property factors the joint forward probability into $P(x_{t,ij} \mid x_{s,ij})P(x_{s,ij} \mid x_0)$. Bayes' rule (Equation 6) divides this joint probability by $P(x_{t,ij} \mid x_0)$ to obtain the bridge probability. Averaging these bridge probabilities over $\tilde{p}_{ij}(x_0 \mid x_{t,ij})$ yields Equation 12.

F.2 Solver Evaluation on TSP-500

This subsection sweeps the multi-step CTMC samplers of Section 3 on TSP-500 under greedy decoding, isolating solver effects from the NEE projection used in Table 1. The canonical Table 1 configuration is 5-step DEIS-2 paired with NEE. All Time entries follow the per-instance batch=1 protocol of Appendix E.2.

Table 10: Solver sweep on TSP-500 with greedy decoding (no NEE, no 2-opt). NFE: number of score-network evaluations. All EDISCO rows share the same trained model. DIFUSCO and T2T are discrete-time references re-measured under the same per-instance, batch=1, single-thread protocol. Bold marks the 5-step DEIS-2 canonical configuration of Table 1; **best** and second-best per Gap and Time column among EDISCO solvers (ties on Time, e.g., DEIS-2-5 and PNDM-5 at 0.11 s, are jointly marked). The DEIS-2-5 row anchors the encoder/decoder-ablation Full entries of Table 5 at 2.10%; PNDM-50 is reported as the slower-but-slightly-better-quality alternative.

Method	**Type**	Steps	NFE	Length ↓	Gap ↓	Time (s) ↓
Concorde [2]	Exact	–	–	16.55	0.00%	18.65
DIFUSCO [78]	SL+G	120	120	18.11	9.41%	2.67
T2T [48]	SL+G	20	∼60	17.39	5.09%	2.30
EDISCO (Euler) [74]	SL+G	50	51	17.08	3.18%	1.01
EDISCO (DEIS-2, canonical configuration) [93]	SL+G	5	6	16.90	2.10%	**0.11**
EDISCO (DEIS-2)	SL+G	10	11	16.88	1.99%	0.20
EDISCO (Heun/RK2) [13]	SL+G	10	20	16.88	1.99%	0.39
EDISCO (RK4) [13]	SL+G	5	20	16.88	1.97%	0.38
EDISCO (DPM-Solver-2) [56]	SL+G	25	50	16.88	2.03%	1.09
EDISCO (PNDM) [55]	SL+G	5	6	17.41	5.29%	**0.11**
EDISCO (PNDM-50)	SL+G	50	51	**16.87**	**1.95%**	1.03

DEIS-2 with 5 steps is the Table 1 canonical configuration because it reaches a competitive greedy gap of 2.10% at 0.11 s per instance, a 24× speedup over DIFUSCO. PNDM-50 attains a slightly lower greedy gap of 1.95% but takes 9.4× longer (1.03 vs 0.11 s per instance), making it impractical as the headline solver. Pairing DEIS-2-5 with NEE in Table 1 collapses the 2.10% greedy gap to 0.018%, recovering the residual quality through the invariant feasibility projection. Higher-order ODE solvers (Heun/RK2, RK4, DPM-Solver-2) approach PNDM-50 quality with intermediate compute. At 50 steps, Euler has a higher gap than PNDM (3.18% vs. 1.95%) at similar runtime (1.01 vs. 1.03 s). The NFE column counts every score-network call. The single-evaluation samplers (Euler, DEIS-2, PNDM) use one call per step plus the final clean-state prediction of Algorithm 2, Heun/RK2 and DPM-Solver-2 use two calls per step, and RK4 uses four, so RK4 at 5 steps costs 20 evaluations, the same as Heun/RK2 at 10 steps. PNDM at 5 steps reaches only 5.29% because the AB-style smoother is unreliable before its history buffer fills, motivating the choice of DEIS-2 in the few-step regime where the smoothing history is short.

F.3 TSP-10000 Results

This appendix extends Table 1 to TSP-10000, the largest TSP scale in the diffusion-CO benchmark literature. TSP-10000 is reported by only a subset of methods because the dense $O(n^2)$ pairwise variables stress memory and compute beyond the reach of most published implementations. EDISCO retains the Table 1 canonical configuration of 5-step DEIS-2 paired with NEE, and the kNN sparsification of Section 4 preserves the invariance argument at this scale.

Table 11: TSP-10000 results. Gap is relative to LKH-3 (default). Time is per-instance wall-clock measured at batch size 1, single-thread, on the hardware described in Appendix E.2; all Time entries are re-measured under this protocol following Wang et al. [84, 85]. **Best** and second-best marked per Gap and Time column among learning-based solvers achieving sub-2% Gap (faster low-quality solvers such as GLOP at 4.90% and AM at 97.17% are excluded from the ranking). RL: reinforcement learning, SL: supervised learning. G: greedy, 2O: 2-opt, S: sampling, AS: active search, AE/NAE/NEE: adaptive / native adaptive / native edge expansion. "–": not included in the matched-hardware comparison.

Method	Type	Length ↓	Gap ↓	Time ↓
Reference Solvers				
LKH-3 [31]	Heuristic	71.77	0.00%	248 s
2-opt [17]	Heuristic	91.16	27.02%	13.4 s
Farthest Insertion	Heuristic	80.59	12.36%	6.2 s
Non-Generative Neural Solvers				
AM [41]	RL+G	141.51	97.17%	3.6 s
GLOP [90]	RL+G	75.29	4.90%	0.89 s
DIMES [71]	RL+AS+G	80.45	12.09%	86.3 s
COExpander [59]	SL+AE+2O	72.80	1.45%	29.49 s
Generative Neural Solvers				
DIFUSCO [78]	SL+G+2O	73.99	3.10%	16.6 s
T2T [48]	SL+S	73.81	2.84%	69.5 s
DISCO [94]	SL+S	73.81	2.84%	22.9 s
Fast-T2T [49]	SL+G+2O	72.90	1.60%	40.93 s
StruDiCO [84]	SL+G+2O	–	–	–
GenSCO [50]	SL	–	–	–
NEXCO [85]	SL+NAE	–	–	–
EDISCO (this work)	SL+NEE	72.79	**1.42%**	**26 s**

Table 11 mirrors Table 1 at $N = 10000$. Among the learning-based solvers evaluated under the matched-hardware protocol with sub-2% Gap, EDISCO leads on Gap (1.42%) and Time (26 s per instance), edging COExpander's adaptive expansion (1.45% at 29.49 s) on each axis and outpacing the diffusion baselines DIFUSCO (3.10%), T2T (2.84%), DISCO (2.84%), and Fast-T2T (1.60%). Faster solvers exist (GLOP at 0.89 s, AM at 3.6 s, DIFUSCO at 16.6 s) but at substantially worse quality (4.90%, 97.17%, 3.10% respectively). NEXCO also reports TSP-10000 results (1.53% Gap in 52 s) in Appendix B.2, Table 8 of Wang et al. [85]. These published figures are reported separately because they were obtained on the source paper's hardware rather than the hardware of Appendix E.2. Classical heuristics 2-opt (27.02%) and Farthest Insertion (12.36%) anchor the upper end of the gap range, framing the difficulty of TSP-10000 for non-learning baselines.

The same speed-quality knob through alternative CTMC solvers (PNDM, Heun/RK2, RK4, DPM-Solver-2) is characterized at TSP-500 in Appendix F.2 and carries over to TSP-10000 without retraining; the canonical DEIS-2-5 + NEE configuration of Table 1 is used here.

F.4 TSPLIB Results

We evaluate EDISCO on real-world TSP instances from the TSPLIB benchmark [72]. Following Hudson et al. [34], Li et al. [48], we train EDISCO on randomly generated 100-node TSP instances and evaluate it on TSPLIB instances ranging from 50 to 200 nodes. To preserve compatibility with published TSPLIB numbers under their canonical decoding configurations, EDISCO is reported here under the prior 4×-sampling + 2-opt protocol; NEE generalization is evaluated separately on the synthetic TSP and CVRP benchmarks of Tables 1 and 3.

Table 12 reports optimality gaps across 29 TSPLIB instances at sizes 51–200. EDISCO leads on aggregate at 0.039%, ahead of GenSCO (0.060%), NEXCO (0.09%), and T2T (0.133%). Older non-diffusion baselines (AM, GCN at autoregressive and supervised-construction settings, Learn2OPT and GNNGLS as 2-opt-based local-search heuristics) have mean gaps of 1.529–40.025% and frame the difficulty of TSPLIB transfer for the older lineage. Each modern diffusion and expansion row is reported under its canonical decoding configuration: DIFUSCO, T2T, Fast-T2T, and EDISCO

Table 12: Optimality gaps (%)↓ for methods trained on random 100-node problems and evaluated on TSPLIB instances with 50–200 nodes. Modern diffusion and expansion methods are re-evaluated on the hardware of Appendix E.2 under each method's canonical decoding configuration. DIFUSCO, T2T, Fast-T2T, and EDISCO use $4\times$ sampling with 2-opt post-processing; GenSCO, NEXCO, and COExpander use greedy with 2-opt. AM, GCN, Learn2OPT, and GNNGLS values are taken from Table 3 of Hudson et al. [34]. **Best** and second-best per row. Means are computed from the 29 displayed instance results.

Instance	AM	GCN	Learn2OPT	GNNGLS	DIFUSCO	T2T	Fast-T2T	GenSCO	NEXCO	COExpander	EDISCO (this work)
eil51	1.628%	8.339%	0.067%	**0.000%**	0.314%	0.314%	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
berlin52	4.169%	33.225%	0.449%	0.142%	**0.000%**	**0.000%**	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
st70	1.737%	24.785%	0.040%	0.764%	0.172%	**0.000%**	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
eil76	1.992%	27.411%	0.096%	0.163%	0.217%	0.163%	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
pr76	0.816%	27.793%	1.228%	0.039%	0.043%	0.039%	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
rat99	2.645%	17.633%	0.123%	0.550%	0.016%	**0.000%**	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
kroA100	4.017%	28.828%	18.313%	0.728%	0.050%	**0.000%**	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
kroB100	5.142%	34.686%	1.119%	0.147%	0.006%	**0.000%**	0.65%	**0.000%**	**0.00%**	0.742%	**0.000%**
kroC100	0.972%	35.506%	0.349%	1.571%	**0.000%**	**0.000%**	**0.00%**	**0.000%**	**0.00%**	0.355%	**0.000%**
kroD100	2.717%	38.018%	0.866%	0.572%	**0.000%**	**0.000%**	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
kroE100	1.470%	26.589%	1.832%	1.216%	**0.000%**	**0.000%**	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
rd100	3.407%	50.432%	0.003%	0.459%	**0.000%**	**0.000%**	**0.00%**	**0.000%**	**0.00%**	0.100%	**0.000%**
eil101	2.994%	26.701%	0.387%	0.201%	0.124%	**0.000%**	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
lin105	1.739%	34.902%	1.867%	0.606%	0.441%	0.393%	**0.00%**	**0.000%**	0.54%	**0.000%**	0.030%
pr107	3.933%	80.564%	0.898%	0.439%	0.714%	0.155%	0.62%	**0.000%**	0.08%	0.100%	0.040%
pr124	3.677%	70.146%	10.322%	0.755%	0.997%	0.584%	0.08%	**0.000%**	0.58%	0.783%	0.075%
bier127	5.908%	45.561%	3.044%	1.948%	1.064%	0.718%	1.50%	**0.000%**	0.66%	**0.000%**	0.220%
ch130	3.182%	39.090%	0.709%	3.519%	0.077%	0.077%	**0.00%**	**0.000%**	**0.00%**	0.597%	0.025%
pr136	5.064%	58.673%	**0.000%**	3.387%	0.182%	**0.000%**	0.01%	**0.000%**	**0.00%**	0.146%	**0.000%**
pr144	7.641%	55.837%	1.526%	3.581%	1.816%	**0.000%**	0.39%	**0.000%**	**0.00%**	**0.000%**	**0.000%**
ch150	4.584%	49.743%	0.312%	2.113%	0.473%	0.324%	**0.00%**	**0.000%**	**0.00%**	0.893%	0.030%
kroA150	3.784%	45.411%	0.724%	2.984%	0.193%	0.193%	**0.00%**	**0.000%**	0.02%	0.100%	0.025%
kroB150	2.437%	56.745%	0.886%	3.258%	0.366%	0.021%	0.07%	0.340%	0.02%	0.191%	**0.013%**
pr152	7.494%	33.925%	**0.029%**	3.119%	0.687%	0.687%	0.19%	0.187%	0.68%	0.191%	0.180%
u159	7.551%	38.338%	0.054%	1.020%	**0.000%**	**0.000%**	**0.00%**	**0.000%**	**0.00%**	**0.000%**	**0.000%**
rat195	6.893%	24.968%	0.743%	1.666%	0.887%	0.018%	0.79%	0.194%	0.11%	0.419%	**0.012%**
d198	373.020%	62.351%	0.522%	4.772%	**0.000%**	**0.000%**	0.86%	0.751%	**0.00%**	0.488%	0.350%
kroA200	7.106%	40.885%	1.441%	2.029%	0.259%	**0.000%**	0.49%	0.160%	**0.00%**	0.056%	0.045%
kroB200	8.541%	43.643%	2.064%	2.589%	0.171%	0.171%	2.50%	0.098%	**0.00%**	0.851%	0.085%
Mean	16.768%	40.025%	1.725%	1.529%	0.320%	0.133%	0.28%	0.060%	0.09%	0.207%	**0.039%**

at sampling-based decoding with 2-opt, and GenSCO, NEXCO, and COExpander at greedy with 2-opt. EDISCO finds the optimum on every small instance (eil51 through eil101, plus pr136, pr144, u159), takes the strict best on kroB150 (0.013%) and rat195 (0.012%), and is the second-best on most remaining rows. The largest EDISCO gaps appear on the harder mid-range instances (d198 0.350%, bier127 0.220%, pr152 0.180%), but every per-instance gap stays below 0.4%, giving the most uniform behavior in the table.

F.5 Noise Schedule Design for TSP Diffusion

We conducted a comprehensive comparison of different noise schedule designs for the continuous-time categorical diffusion model applied to TSP. The noise schedule $\beta(t)$ controls the rate of information destruction during the forward diffusion process and significantly impacts model performance.

We evaluated three families of noise schedules:

Linear Schedule: Following the standard approach in diffusion models [32], we tested linear schedules with:

$$\beta(t) = \beta_{\min} + t(\beta_{\max} - \beta_{\min}). \tag{21}$$

Exponential Schedule: Based on Campbell et al. [14], we evaluated exponential schedules:

$$\beta(t) = ab^t \log(b). \tag{22}$$

Cosine Schedule: Following Sun et al. [77], we tested cosine schedules with improved numerical stability:

$$\beta(t) = \text{clip}\left(\frac{\pi}{4} \cdot \frac{\tan(\pi t/2)}{\sqrt{\cos(\pi t/2) + \epsilon}}, \beta_{\min}, \beta_{\max}\right). \tag{23}$$

We trained each schedule variant for 50 epochs on TSP-50 using 10,000 randomly generated instances for fast verification. The same network architecture and hyperparameters were maintained across all experiments. Each schedule family was tested with three different parameterizations: baseline (standard parameters), conservative (slower noise injection with smaller β values), and aggressive (faster noise injection with larger β values).

Table 13: Comparison of noise schedules for TSP diffusion on TSP-50. All models trained for 50 epochs with 50 diffusion steps. Bold indicates the lowest gap and inference time.

Schedule Type	Configuration	Optimality Gap (%)↓ Best	Optimality Gap (%)↓ Final	Conv. Epoch↓	Inference Time (s)↓
Linear	Aggressive: $\beta \in [0.1, 2.0]$	2.88	3.03	50	1.06
	Baseline: $\beta \in [0.1, 1.5]$	**2.29**	**2.63**	45	1.06
	Conservative: $\beta \in [0.1, 1.0]$	2.74	3.51	50	1.16
Exponential	Baseline: a=0.5, b=4.0	3.39	3.60	50	**1.04**
	Conservative: a=0.3, b=3.0	4.19	4.42	50	1.13
	Aggressive: a=0.8, b=5.0	2.60	3.78	50	1.05
Cosine	Aggressive: $\beta \in [0.001, 10.0]$	4.83	5.37	45	1.05
	Baseline: $\beta \in [0.01, 5.0]$	3.45	4.51	40	1.07
	Conservative: $\beta \in [0.1, 3.0]$	3.25	4.09	40	1.06

Table 13 presents the comprehensive results. Linear schedules demonstrated the best overall performance, with the baseline configuration ($\beta_{\min} = 0.1, \beta_{\max} = 1.5$) achieving the lowest validation gap of 2.29%. The aggressive variant ($\beta_{\max} = 2.0$) and conservative variant ($\beta_{\max} = 1.0$) both underperformed at 2.88% and 2.74% respectively, supporting moderate noise injection in this TSP-50 experiment.

Exponential schedules showed high sensitivity to parameter selection, with best validation gaps ranging from 2.60% to 4.19% and final gaps ranging from 3.60% to 4.42% across configurations. Cosine schedules consistently underperformed with gaps ranging from 3.25% to 4.83%, indicating that their non-linear noise profile is less effective in this TSP-50 experiment.

The superiority of linear schedules in TSP contrasts with image generation, where cosine schedules often excel [63]. We attribute this to the discrete nature of TSP adjacency matrices, which benefit from uniform, predictable noise injection rather than variable rates. These findings validate our choice of $\beta_{\min} = 0.1, \beta_{\max} = 1.5$ for all experiments, based on the tested TSP-50 configurations.

F.6 Evaluation on Architectural Hyperparameters

We conduct systematic evaluations of critical architectural hyperparameters to justify our design choices.

Step Size α for Coordinate Updates The step size α in Equation 8 controls the magnitude of coordinate updates during message passing. We evaluate different values on TSP-50:

Table 14: Effect of step size α on model performance and training stability (TSP-50). *Standard deviation of coordinates after 8 layers (optimal range: 0.3-0.4).

Step Size α	0.01	0.05	**0.1**	0.2	0.5
Optimality Gap (%)↓	0.08	0.03	**0.01**	0.15	Diverged
Coordinate Std*	0.42	0.38	**0.35**	0.18	0.02
Training Stable	✓	✓	✓	Unstable	Collapsed
Convergence Epoch↓	42	38	**35**	48	-

As shown in Table 14, smaller α values (0.01, 0.05) maintain stability but converge more slowly and achieve suboptimal performance. Larger values ($\alpha \geq 0.2$) cause instability and coordinate contraction, with the coordinate standard deviation falling to 0.18 at $\alpha = 0.2$ and 0.02 at $\alpha = 0.5$, where training collapses.

Temperature Parameter τ for Weight Scaling The temperature parameter τ in $w_{ij} = \tanh(\text{MLP}_c(\mathbf{m}_{ij})/\tau)$ scales the MLP outputs before applying tanh, preventing gradient saturation:

Table 15 shows that $\tau = 10$ achieves the optimal balance. Lower values ($\tau \leq 5$) cause excessive saturation, leading to vanishing gradients and slower convergence. Higher values ($\tau \geq 20$) reduce the non-linearity's effectiveness, diminishing the model's expressiveness.

Table 15: Effect of temperature τ on gradient flow and performance (TSP-50). †Average gradient norm in coordinate MLP during first 10 epochs. ‡Percentage of tanh outputs with $|w_{ij}| > 0.95$.

Temperature τ	1	5	**10**	20	50
Optimality Gap (%)↓	0.12	0.04	**0.01**	0.02	0.05
Avg. Gradient Norm†	0.003	0.018	**0.042**	0.038	0.031
Tanh Saturation Rate‡	68%	24%	**8%**	12%	18%
Convergence Epoch↓	52	40	**35**	36	39

Joint Impact Analysis We evaluate the joint effect of α and τ on TSP-100 performance:

Table 16: Joint ablation of α and τ on TSP-100 (optimality gap %↓).

α \ τ	1	5	10	20	50
0.01	0.21	0.15	0.08	0.09	0.12
0.05	0.18	0.09	0.05	0.06	0.08
0.1	0.15	0.06	**0.04**	0.05	0.07
0.2	Unstable	0.18	0.15	0.16	0.19
0.5	Collapsed	Collapsed	Diverged	Diverged	Diverged

The joint ablation confirms that $\alpha = 0.1$ and $\tau = 10$ provide the best tested configuration, with larger step sizes leading to instability or divergence.

Discussion on Hyperparameter Robustness While our ablation studies reveal sensitivity to α and τ during initial design exploration—particularly coordinate contraction and instability when $\alpha \geq 0.2$—we emphasize three important aspects of these architectural stabilizers: (1) **Optimal settings are robust**: Once identified, $\alpha = 0.1$ and $\tau = 10$ work consistently across all problem sizes and types (TSP-50/100/500/1000, CVRP-50/100/200/500, ESTP-10/20/50) without per-problem retuning, as demonstrated throughout our experiments. (2) **Comparable to standard practice**: This sensitivity is analogous to other architectural hyperparameters in deep learning (e.g., learning rates, attention heads, batch normalization momentum), which require careful tuning during model development but remain stable once configured. (3) **Strong generalization validates stability**: Our cross-distribution evaluation (Table 2 in the main text) demonstrates that these settings generalize well to out-of-distribution instances (Cluster, Explosion, Implosion), confirming they are not narrowly tuned to specific data distributions.

The thorough sensitivity analysis presented here provides practitioners with clear guidance for implementation and demonstrates our commitment to transparency in reporting both the strengths and limitations of our architectural design.

F.7 Model Efficiency

Although the results in the main text are from the full-scale EDISCO model, it is interesting to see EDISCO's performance under reduced model sizes. Table 17 presents a comprehensive comparison of model efficiency across different architectures and scales, demonstrating how EDISCO's equivariant design enables strong performance even with reduced model capacity and training data.

EDISCO-Full (5.5M parameters) uses one-third as much training data as baselines on small instances (500K vs 1.5M) and about half as much on large instances. It achieves 0.01% gap on TSP-50 compared to DIFUSCO's 0.48%, and 1.95% on TSP-500 versus DIFUSCO's 9.41%. On TSP-10000, EDISCO-Full achieves 1.98% gap with 3K training instances, outperforming DIFUSCO's 8.95% gap with 6.4K instances.

EDISCO-Medium (1.9M parameters), with about one-third of the parameters of baseline models, achieves 0.03% gap on TSP-50 and 2.18% on TSP-500. It matches EDISCO-Full's performance on TSP-1000 at 2.85% gap. Compared to T2T, which achieves 0.04% on TSP-50 with 5.3M parameters,

Table 17: Model efficiency and performance comparison across TSP scales. Training instances shown in thousands (K) with corresponding optimality gaps (%). All EDISCO entries use 50-step PNDM with greedy decoding (no NEE, no 2-opt) so capacity differences across architectures are isolated from sampler and decoder effects; the main-table SL+NEE configuration of Table 1 therefore reaches lower gaps than the EDISCO-Full row here. EDISCO-Full uses 12 EGNN layers with 256 hidden dimension (5.5M parameters), EDISCO-Medium uses 12 layers with 128 hidden dimension (1.9M parameters), and EDISCO-Small uses 8 layers with 128 hidden dimension (1.3M parameters). †12-layer GNN with width 256. *Same architecture as DIFUSCO.

Method	Model Parameters	Training Instances (K) / Optimality Gap (%)↓				
		TSP-50	TSP-100	TSP-500	TSP-1000	TSP-10000
DIFUSCO†	5.3M	1500 / 0.48	1500 / 1.01	128 / 9.41	64 / 11.24	6.4 / 8.95
DISCO†	5.3M	1500 / 0.16	1500 / 0.58	- / -	- / -	6.4 / 2.90
T2T*	5.3M	1500 / 0.04	1500 / 0.18	128 / 5.09	64 / 8.87	6.4 / 2.92
EDISCO-Full	5.5M	500 / **0.01**	500 / **0.04**	60 / **1.95**	30 / **2.85**	3 / **1.98**
EDISCO-Medium	1.9M	500 / 0.03	500 / 0.08	60 / 2.18	30 / **2.85**	3 / 2.43
EDISCO-Small	1.3M	**300** / 0.08	**300** / 0.7	**40** / 4.18	**20** / 5.21	**2** / 3.18

EDISCO-Medium achieves comparable performance (0.03%) with 1.9M parameters and the same 500K training instances used by EDISCO-Full.

EDISCO-Small (1.3M parameters) uses about 24% as many parameters as baselines and requires the least training data: 300K for TSP-50/100, 40K for TSP-500, and 2K for TSP-10000. It achieves 0.08% gap on TSP-50 and 0.7% on TSP-100. On TSP-500, with 40K training instances, it achieves 4.18% gap, compared to DIFUSCO's 9.41% with 128K instances. On TSP-10000, EDISCO-Small achieves 3.18% gap, which outperforms DIFUSCO (8.95%) and is comparable to DISCO (2.90%).

The results for the amount of training data represent the minimum numbers required to reach the reported EDISCO gaps. After this, even with increased training data, there is no noticeable improvement in the optimality gaps.

F.8 Training Resource Requirements

Table 18 presents comprehensive training resource requirements for EDISCO-Full, demonstrating its efficiency on commodity hardware. EDISCO trains on a single NVIDIA RTX A6000 GPU, using graph sparsification and fewer training instances. The combination of E(2)-equivariance and continuous-time diffusion enables 3× better sample efficiency (500K vs 1.5M training instances for TSP-50/100), reducing the required number of training instances. Measured training time and peak memory are reported below.

Table 18: Training resource requirements for EDISCO-Full on a single NVIDIA RTX A6000 GPU (48 GB). Peak memory usage includes model parameters, optimizer states, and batch processing. EDISCO achieves state-of-the-art performance using 33–50% of the training data of baseline methods while training on a single GPU.

Problem Scale	**Dataset Size**	**Batch Size**	**Training Epochs**	**Training Time↓**	**Peak Memory↓**
TSP-50	500K	64	100	18h 45m	19.2 GB
TSP-100	500K	32	100	68h 30m	26.8 GB
TSP-500	60K	16	50	31h 15m	23.5 GB
TSP-1000	30K	8	50	61h 20m	39.2 GB
TSP-10000	3K	4	50	17h 40m	44.8 GB

Graph sparsification (kNN with $k = 50$ for TSP-500 and $k = 100$ for TSP-1000/10000) reduces memory complexity from $O(n^2)$ to $O(kn)$, enabling practical training on a single GPU even for TSP-10000. TSP-500/1000/10000 use curriculum learning initialized from smaller problem checkpoints, further improving sample efficiency. Peak memory remains within the A6000's 48 GB capacity across all reported TSP scales.

F.9 Visualization of EDISCO's E(2)-Equivariance

Figure 6 illustrates the denoising process of EDISCO on a standard TSP-100 instance. To illustrate the E(2)-equivariant behavior of the network qualitatively, we present the same denoising process on rotated coordinate sets in Figures 7 and 8. Only the greedy decoder is used without any other post-processing techniques. The visualization shows progression from pure noise ($t = 1.0$) to clean tours ($t = 0.0$) across five independent sampling rounds. The rotated coordinate sets are renormalized to the unit square for display, so the per-figure tour lengths reflect each panel's display scale rather than absolute coordinate length; the structural correspondence of the denoising trajectories is what illustrates the rotation-invariant representations established by Corollary 2, with the formal length-invariance argument in Appendix B.2.

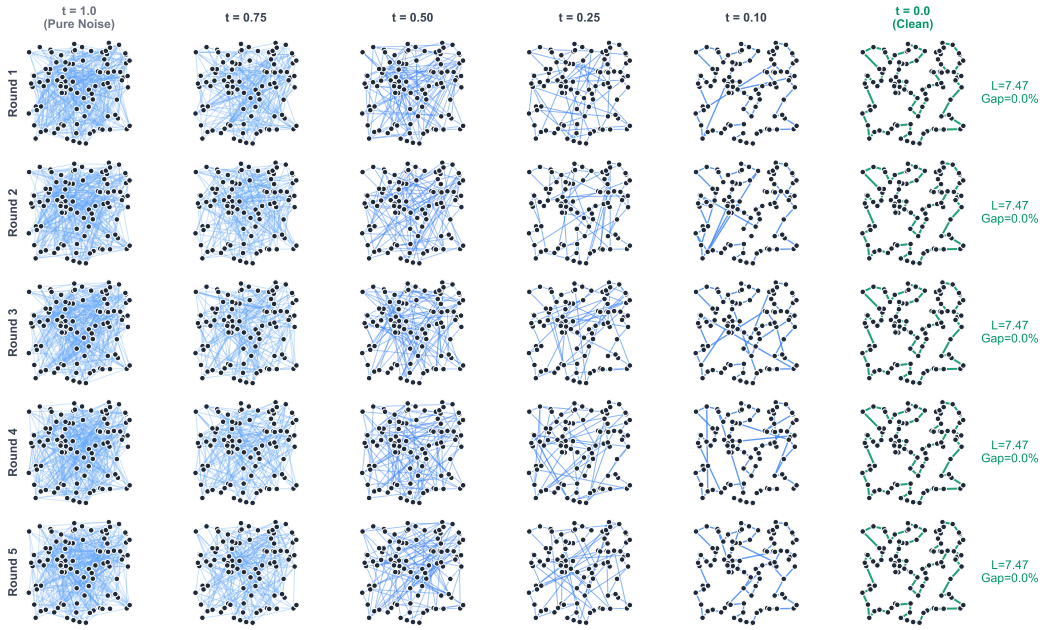

Figure 6: Five rounds of the denoising process of EDISCO on the original TSP instance.

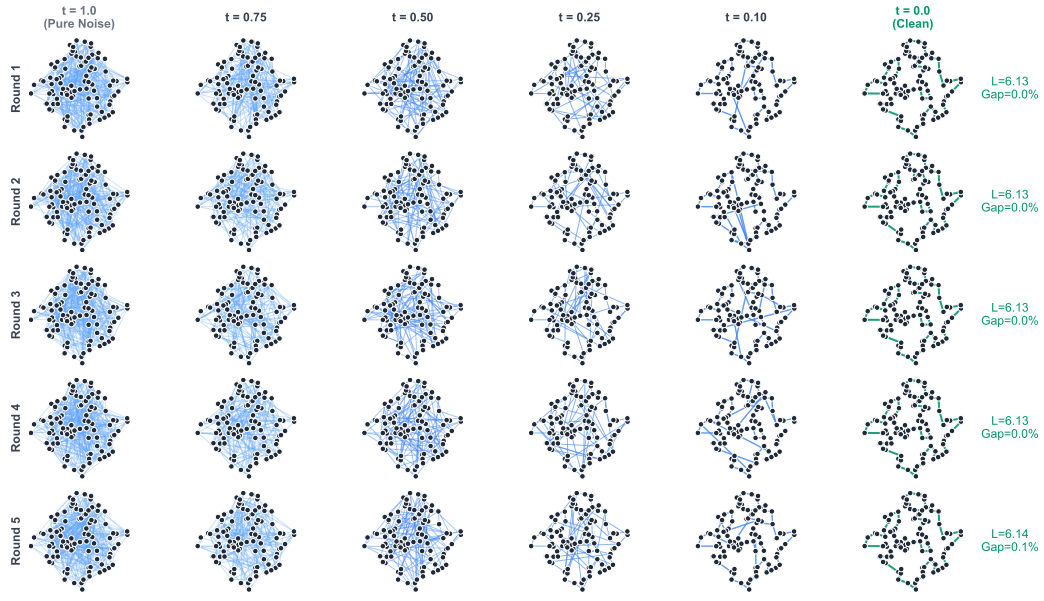

Figure 7: Five rounds of the denoising process of EDISCO on the 45° rotated instance.

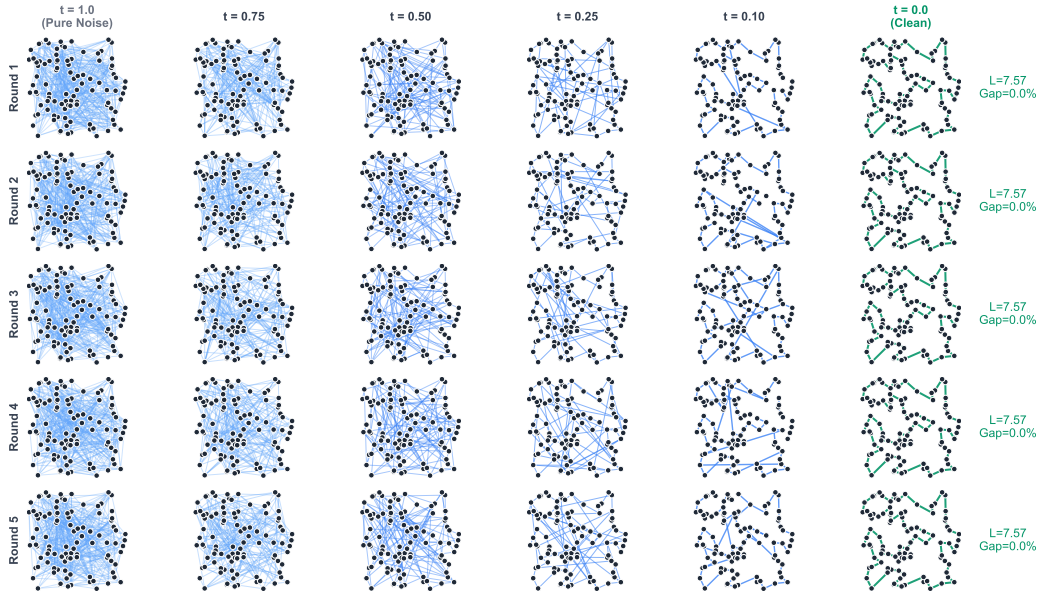

Figure 8: Five rounds of the denoising process of EDISCO on the 90° rotated instance.

F.10 Compute and Runtime Conventions for Recent Competitors

This appendix documents the compute and runtime conventions used for the recent-competitor entries (StruDiCO, GenSCO, NEXCO, COExpander) in Table 1 of Section 4.2. The compute axis differs by method: number of denoising steps for diffusion methods, number of search-driven generation cycles for GenSCO [50], and iteration depth of the adaptive expansion for COExpander [59]. EDISCO is reported at both PNDM-50 and DEIS-2-5 configurations. The Table 1 snapshot uses the minimum budget at which each method achieves its reported gap.

Compute budget conventions. All wall-clock times are per-instance, measured at batch size 1 in single-thread mode on the hardware described in Appendix E.2, following the protocol of Wang et al. [84, 85]. Each method is run at its own canonical decoding configuration (denoising step counts for diffusion methods, search cycles for GenSCO, expansion iterations for COExpander), with timings following the stated measurement conventions and computational budgets remaining method-specific.